MR9401V2

EOSDIS Core System Communications and Systems Management Architecture

White Paper
Working Paper

April 1994

Prepared Under Contract NAS5-60000

RESPONSIBLE ENGINEER

Carl Wheatley /s/ 4/29/94

Carl Wheatley, CSMS Chief Engineer April 29, 1994
EOSDIS Core System Project

SUBMITTED BY

E. M. Lerner /s/ 4/29/94

Ed Lerner, CSMS Office Manager April 29, 1994
EOSDIS Core System Project

Hughes Applied Information Systems, Inc.
Landover, Maryland

This page intentionally left blank.

Contents

1. Introduction

2. Background

3. Communications Sub-Architecture

4. Internetworking Sub-Architecture

5. System Management Sub-Architecture

Figures

Tables

Appendix A. Open Distributed

Appendix B. OSF Distributed

Appendix C. SNMP Management Notes

Appendix D. Notes on ISO/CCITT

Appendix E. Notes on GNMP

This page intentionally left blank.

1. Introduction

1.1 Purpose

The purpose of this document is to present a detailed end-system architecture of the Communications and System Management Segment (CSMS) of the EOSDIS Core System (ECS). It is based on key components of the conceptual architecture presented at the EOSDIS Progress Review, held December 13-14, 1993 in Landover, Maryland.

The Communications and Systems Management Architecture focuses on those portions of the system involved in the interconnection of users and service providers, transfer of information, and system management of ECS components. It supports and interacts with, portions of the Science Data Processing Segment (SDPS) and the Flight Operations Segment (FOS). Those segments are referenced herein only to the extent to establish a context for architectural discussion.

The key objectives of this document are:

- to place CSMS within the context of ECS, the Flight Operations Segment (FOS) and Science Data and Processing Segment (SDPS)
- to identify the relevant framework models for CSMS logical architectures
- to present the CSMS implementation architectures, refining to sets of services
- to identify and discuss relevant Policy issues affecting the CSMS architectures
- to identify and discuss Quality of Service issues, including performance and availability
- to identify and highlight the evolvable nature of the architecture

This document is intended to be used as a source of derivative material in furthering the system-level design of the ECS, and CSMS in particular, for continuing system design efforts leading to the System Design Review (SDR) in June 1994. As a result, support and update of this document past SDR is not planned.

1.2 Organization

This paper is organized as follows:

- Section 1 presents the purpose of the document, its organization, and logistics concerning its review, including scope and context and points of contact.
- Section 2 establishes the context of the Communications and Systems Management Architecture, including its background and origins and a list of relevant resources.
- Section 3 presents the logical and implementation architectures for the Communications Sub-Architecture

- Section 4 presents the logical and implementation architectures for the Internetworking Sub-Architecture
- Section 5 presents the logical and implementation architectures for the System Management Sub-Architecture

1.3 Review and Approval

This document is an informal contract deliverable approved at the Office Manager level. It does not require formal Government review or approval; however, it is submitted with the intent that review and comments will be forthcoming.

This is the second release of the CSMS architecture working paper. It provides the necessary background to develop an aggregation of CSMS services based on known object services and available COTS infrastructural components, and describes an architectural direction. Phased-development plans of the services is not discussed within this document in order to maintain separation of design-related aspects of the system from the architecture. It should be noted that the development of the reference architecture is proceeding in parallel with system design activities. The architecture presented in this document, therefore, represents work in progress and is subject to change.

This revision provides the background and architecture context for the program in support of the baseline system architecture review. It will serve as contextual material for the System Design Specification, DID 207, which will be issued in conjunction with the System Design Review.

Questions regarding technical information contained within this Paper should be addressed to the following ECS and/or GSFC contacts:

- ECS Contact

  Ed Lerner
  CSMS Manager
  (301) 925-0303
  edle@eos.hitc.com

- GSFC Contact

  John Gainsborough
  CSMS Technical Manager
  (301) 286-2153
  johng@ulabsgi.gsfc.nasa.gov

Questions concerning distribution or control of this document should be addressed to:

Data Management Office
The ECS Project Office
Hughes Applied Information Systems, Inc.
1616A McCormick Dr.
Landover, MD 20785

2. Background

2.1 Architecture Definition Process

The architecture process within this document builds on the architecture direction set following the ECS System Requirements Review (SRR), held September 13-14, 1993 in Greenbelt, Maryland, through to the ECS December Progress Review (DPR) held December 13-14, 1993 in Landover, Maryland. An important concept introduced at the DPR was the creation of chronologically-ordered conceptual, logical, and implementation architectures.

This paper documents the logical and implementation architectures for CSMS. Within this document are several references to differing dimensions of architectural analysis. The architecture analysis can be viewed as multidimensional, proceeding in parallel tracks. Figure 1 illustrates the CSMS architectural dimensions of analysis.

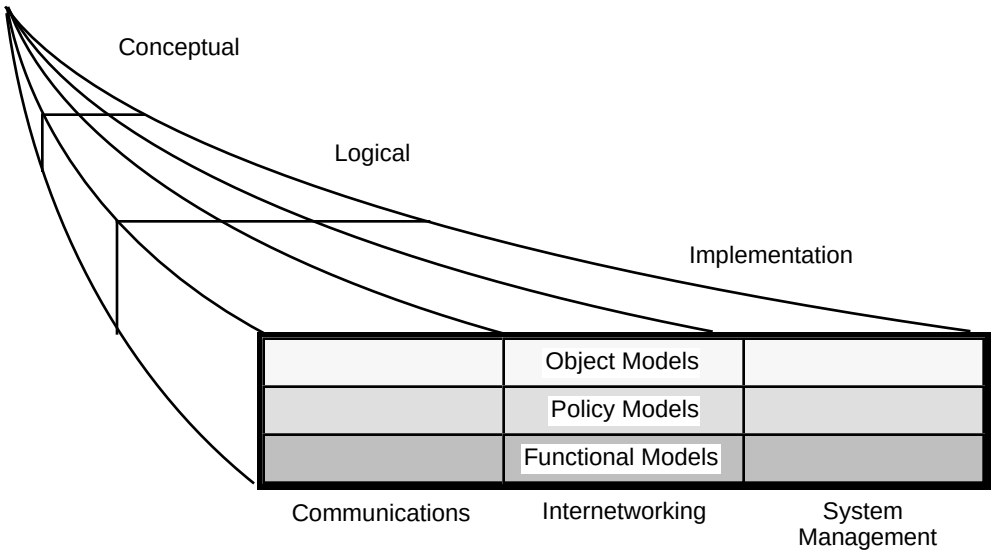


Figure 1. Dimensions of Architectural Analysis

The primary dimension of architectural analysis is chronologically-oriented, proceeding with time into increasing levels of analysis toward integration to the system design phase. Key phases of this architectural dimension of analysis include the conceptual, logical, and implementation architectures. The conceptual architecture is based on the high-level system model presented at the

December Progress Review (DPR). This architecture is discussed in the EOSDIS Core System Science Information Architecture White Paper and is not repeated here. At the DPR, the drivers and key architectural principles were presented, followed by a discussion of organizational and functional descriptions of the ECS. The conceptual architecture presented at DPR succeeded in defining the *goals and objectives* for the ECS architecture.

The logical architecture is the selection of reference model(s), that have the essential behavior and attributes that the implementation architecture and system-level design should inherit to meet the system objectives as laid out by the conceptual architecture. In essence, the logical architecture defines the *strategy* of how the CSMS architecture will be further advanced. The implementation architecture is the *tactical* level of architectural analysis, defining major services within each of the logical architecture components.

The second dimension of architectural analysis is to effect concurrency within the analysis effort. Parallelism in analysis is introduced by the early decomposition of the architecture into sub-architectures. The logical architecture analysis is integrated with sub-architecture analysis work to break CSMS into major logical components, or sub-architectures, that can be analyzed in parallel with minimal functional overlap and interdependency to other sub-architectures, including SDPS and FOS sub-architectures. For CSMS, the sub-architectures include the communications, internetworking, and system management sub-architectures. Each sub-architecture section includes a logical and implementation architecture discussion.

The third dimension of architectural analysis provides perspective models. These models are used to help describe the logical and implementation architectures. The primary representation model is the object model. The object model is an ongoing tool for analysis that is refined through multiple iterations of generalization/specialization to fully inherit the goals/objectives/strategy for the CSMS-ECS. A highly aggregated form of the object model is used to define services that are provided by each CSMS sub-architecture for external use. The object model is performed using the OMT tool and the GE-Rumbaugh notation. A simplified notation of the GE-Rumbaugh notation adopted by OMT is used in the many figures of this document to illustrate system, segment, and sub-architecture context and services.

The early beginnings of a second important perspective model is provided textually - the policy model. The architecture is developed to be policy neutral, however, the policy model defines key aspects of the CSMS architecture-driven design that will ultimately require policy decision for implementation. Policy implications are discussed in appropriate places within this document for each sub-architecture.

A third model, included within this document, are functional models and graphics. These representations of the architecture are used to help illustrate alternative implementations, and where required, to facilitate the behavioral aspects of collections of objects and services.

Additional models that allocate the object model to physical entities to aid in the analysis of system performance studies are not defined within this document. This area of modeling will be reexamined post-SDR as ECS draws deeper into the design stage.

2.2 Related documents

This document focuses on the CSMS logical and implementation architectures, and the supporting rationale, with minimal background discussion. It assumes familiarity with the concepts developed in related project documents, including:

193-00646	ECS Reference Models, December 1993
193-00561	DCE Migration Study
193-00611	Science-based System Architecture Drivers for the ECS Project White Paper, December, 1993
193-00136	Version 0 Data Migration and Translation Tool Analysis
193-00623	ECS Evolutionary Development White Paper, December, 1993
193-000626	GCDIS/UserDIS Study White Paper, January 1994
193-00632	DME Migration Study
FB9402V1	ECS Science Requirements Summary White Paper, February 1994
FB9401V1	ECS Science Information Architecture White Paper, February 1994
604/OP1	ECS Operations Concept (revision TBD)
207/SE1	SDS Outline and SDR Agenda (TBD)

2.3 ECS Context

The context diagram for ECS is shown in Figure 2. The ECS provides many services for both internal and external users through a distributed processing architecture. The CSMS is shown within the ECS context as providing communications, internetworking, and systems management capabilities.

2.4 CSMS Context and Sub-Architecture Divisions

The CSMS context, within the ECS, is shown in Figure 3. Figure 4 illustrates the first-tier decomposition of CSMS. The major sub-architectures of CSMS are identified in Figure 4 include the communications, system management, and internetworking sub-architectures. The sub-architectures were selected on the basis of the supporting logical architecture frameworks to be presented. The sub-architectures provide minimal interdependencies to each other and other sub-architectures of the FOS and SDPS.

Major FOS and SDPS sub-architecture behavioral and performance dependencies are directly with the communications sub-architecture. The communications sub-architecture is the applications and user access path to all ECS services.

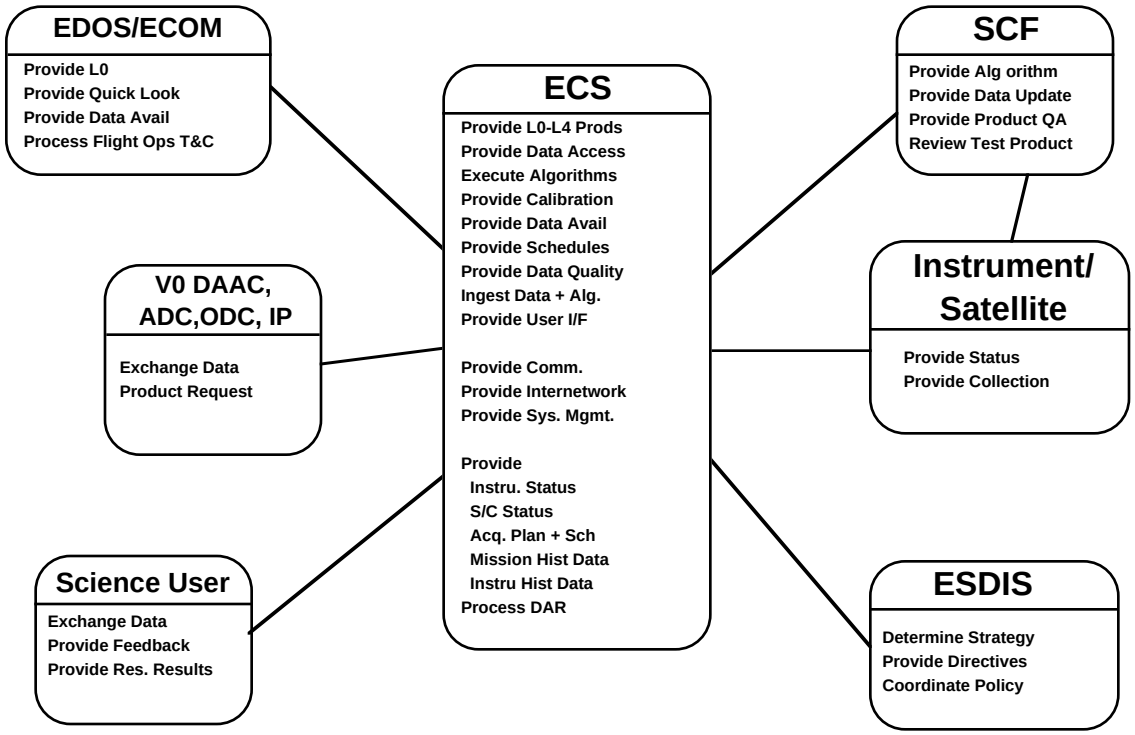


Figure 2. ECS Object Model Context

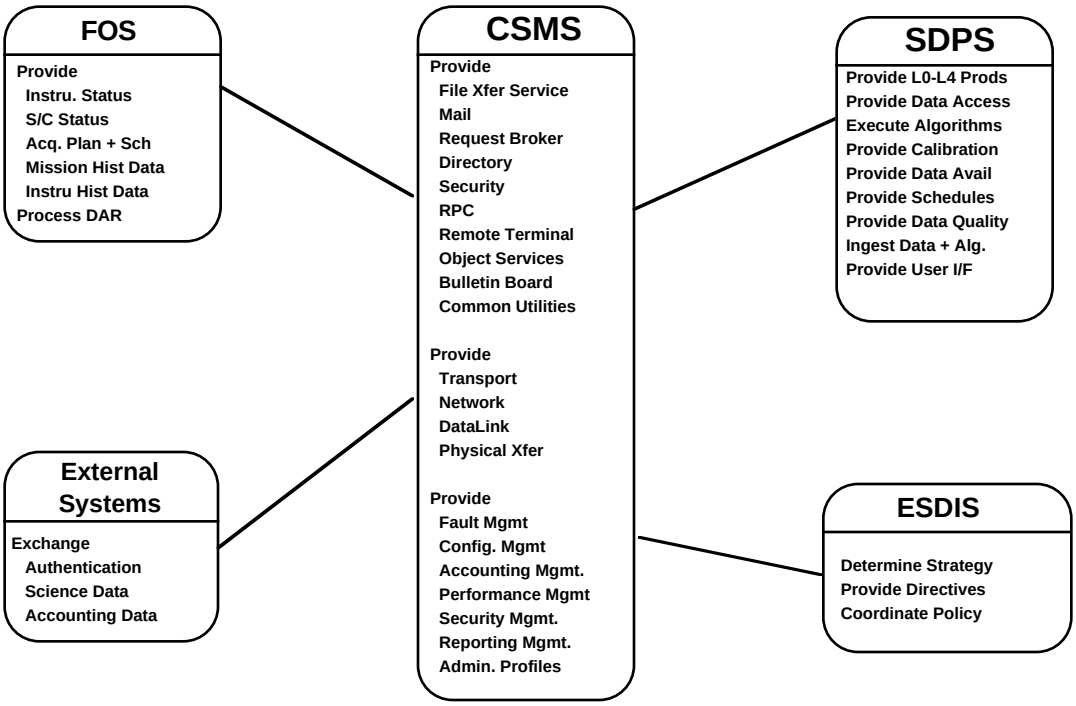


Figure 3. CSMS Object Model Context

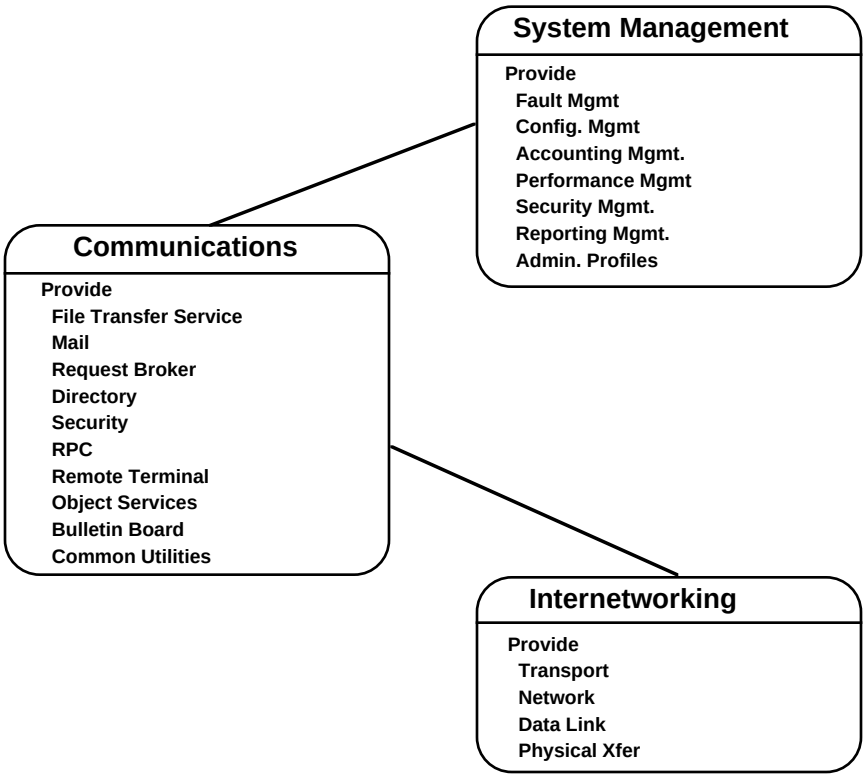
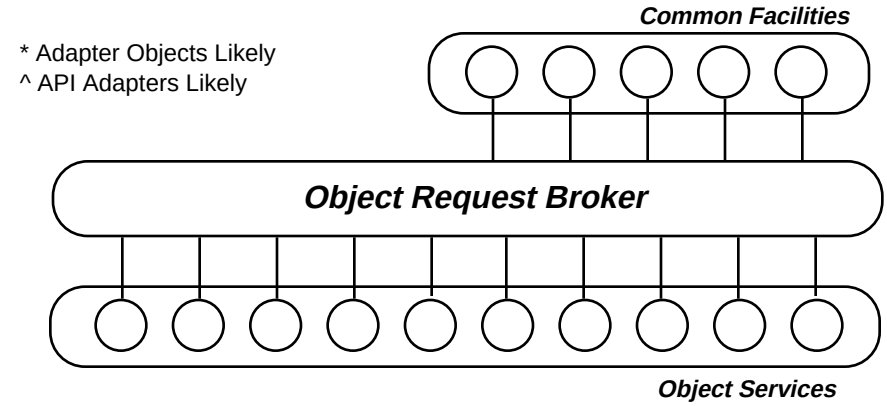


Figure 4. CSMS Sub-Architecture Decomposition

2.4.1 Communication Sub-Architecture

Primary services of the communications sub-architecture are depicted in Figure 5. From an open systems interconnection-reference model (OSI-RM, or ISO 7498:1994, Open Systems Interconnection) perspective, the communications sub-architecture is comprised of layers 5-7, the session, presentation, and applications layers. Support in this sub-architecture area is provided for peer-to-peer, advanced distributed, messaging, management, and event-handling communications facilities. These services typically appear on communicating end-systems across an internetwork. Additionally, services within OSI-RM layers 5-7 to support communicating entities are provided, included directory, security, time, and other ancillary services. The services of the communications sub-architecture are functionally dependent on the services of the internetworking sub-architecture. Many of the common facilities may be jointly developed by CSMS with the other segments.



Object Services	***Object Services (Cont.)***	**Common Facilities**
Mission Critical	*Mission Essential (Cont.)*	Object Collections
Security*^	Transaction	Object Instantiation
Naming*^	Concurrency	Management*^
Event	Externalization	Cataloging and Browsing
I/F Repository	Change Management	Help Facilities
Implementation Repository	Replication	Printing and Spooling
	Threads	Information Search and Retrieval
Mission Essential	Time	Agents
LifecyclePersistence	Archive	Object Interchange
Relationships	Backup/Restore	Clipboard
Data Interchange*	Startup	Policy Management
Licensing	Installation/Activation	Distributed Object Manager
Trading*	Operational Control	Load Balancing
Query*		File Access*^
Properties		Electronic Mail^
		Telnet^

Figure 5. Communications Services

2.4.2 Internetworking Sub-Architecture

Primary services of the interworking sub-architecture are depicted in Figure 6. From an OSI-RM perspective, the internetworking sub-architecture is comprised of layers 1 through 4 - the physical, data link, network, and transport layers. Support in this sub-architecture area provides alternative transports between communicating end-stations, alternative networking methods between end systems and intermediate systems, and alternative circuit, packet or cell-based LAN and WAN distribution services. The internetworking architecture services are not functionally dependent on any services outside of themselves.

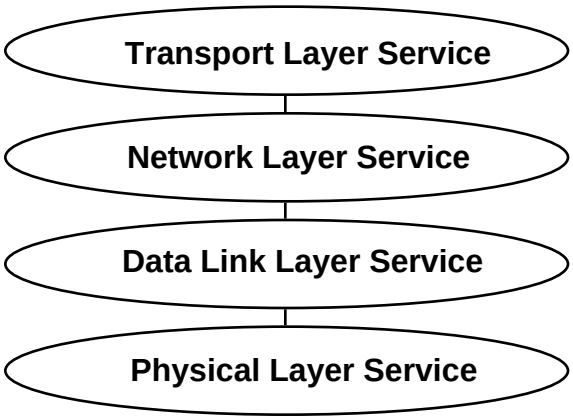
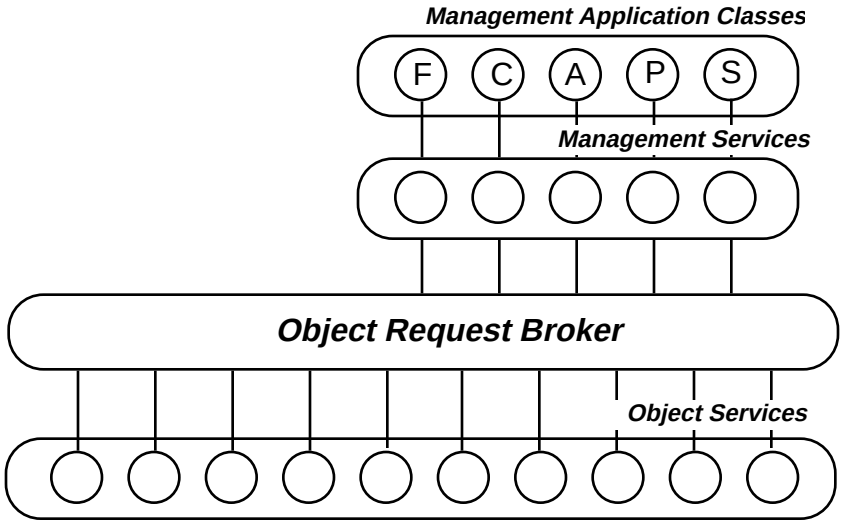


Figure 6. InternetworkingServices

2.4.3 System Management Sub-Architecture

Primary services of the system management sub-architecture are depicted in Figure 7. The system management sub-architecture is primarily in the application domain, above the OSI-RM application-layer services, as management applications classes. Management services and object services support the management application classes. Managed object classes for networks and protocol state machines are a part of the system management sub-architecture information model (not shown), while protocol mechanisms for the *transfer* of system management information are discussed as part of the communication sub-architecture. Like services within FOS and SDPS, the system management sub-architecture services are functionally dependent on the services of both the communications and internetworking sub-architectures.



Object Services	**Management Services**	**Management App. Class**
Mission Critical	MUI	Configuration Management
Lifecycle	Management*^	Performance Management
Naming*^	Maps/Collections	Accounting Management
Event	Monitoring	Software Management
Persistence	Discovery	Fault Resolution Management
Security*^	Agents	Security Management
Relationships	Policy-Driven Base	Printer Management
	Policy Mgmt	Backup and Restore
Mission Essential	Instance Mgmt.	
Transaction	Scheduling Mgmt.	
Concurrency	Cataloging and Browsing	* Adapter Objects Likely
Time	Customization	^ API Adapters Likely

Figure 7. System ManagementServices

3. Communications Sub-Architecture

3.1 Assessment of Needs/Drivers

In response to the SRR, many scientists have been visited through a series of field trips. As part of that effort, the key technology drivers to satisfy the scientist expectations were assessed. The key technology drivers for the communications infrastructure are very broad in scope, supporting a maximum of flexibility for interoperability, transparency, and heterogeneity for distributed processing, evolving toward collaborative processing. The list of the technology drivers and their importance is summarized in Table 1.

The goal in striving to achieve these technology drivers is to support advanced heterogeneous distributed processing, with support for interoperability and transparency. The architecture needs to be built to provide extensibility to support these drivers, both for support to GCDIS/UserDIS, and for evolutionary advances toward fully collaborative processing.

Support for most of the technology drivers is available today, although not all to the degree required for a high-performance and high-capacity system like ECS. Each of the respective technologies in the above list is being tracked by the ECS team - in many cases, we are actively involved in the promotion of the activities that contribute to the technologies development.

The scientist visits and subsequent discussions have led to the identification of several specific needs for the ECS communications sub-architecture. The communications sub-architecture must address the following:

- how to support advances in infrastructure and interoperability layers
- how to support multiple protocols of similar function and support the evolution of protocols
- how to accept new services into the system that are not developed in parallel with the client code
- how clients can gain information about new services, and then access the service
- how to provide support for the mixing and matching of available services
- how to provide support for legacy system integration within ECS
- how to protect against unwanted Application Programming Interface (API) changes throughout the ECS Lifecycle

3.2 Logical Architecture

3.2.1 Framework Approach

A major goal to achieving interoperability among open systems is to make applications interoperable across a heterogeneous environment. For an application to be interoperable in a

heterogeneous environment, it must have implementation-specific dependencies removed. Service interfaces to client processes, done correctly, can protect applications from the specifics of network protocols, network-based services, user interfaces, operating systems, and data stores.

Table 1. Importance of Technology Drivers

Technology Driver	Importance
Synchronous Interprocessing	Basic Distributed Processing Requirement
Asynchronous Messaging	Provides Efficient Utilization of Client Resources
Static Invocation	Another Basic Distributed Processing Requirement
Explicit Static Binding	Enables Direct Client Access to Remote Services
Implicit Static Binding	Enables Client Access to Services without A-Priori Knowledge of Service Location
Directory Service Scalability	Provides Graceful Growth of Directory Elements through ECS Lifecycle
Naming Service	Enables Extension/Abstraction of Directory Service for Identification of Remote Objects
Security Service	Inhibits Unwanted Intruders - Protects Resources
Object Technology	Essential Enabler for Advanced Distributed and Collaborative Processing
Time Synchronization	Essential for correlation of events across sites
Multivendor Interoperability	Provides Flexibility and Evolvability over ECS Lifecycle
O/S Transparency	Enables Technology Migration to Advanced O/Ss
Event Processing	Provides Efficient Distributed Processing of All Events and Error Conditions
Concurrency	Increases Utilization of System Resources
Internationalized Security	Enables International Security Extensions
Multiple Language Support	Provides Application Developer Flexibility and Support for Legacy Code
Legacy Server Integration	Enables Reuse of Existing Applications with Minimal Transition Difficulty
Dynamic Invocation	Enables Client Access to Services without A-Priori Knowledge of the Service Existence and Service Access Operations
Dynamic Load Balancing	Provides Efficient Assignment of System Resources and Improves Performance
Request Brokering	Essential Enabler to Carry Out Distributed Processing Operations with Minimal Client Burden
Server Export/Scaling	Provides for scientist advertising of services
Real-Time Collaboration	Enables Scientist to Scientist Direct Interaction
Trading	Primary Client Interface for Service Negotiation (Import) and Naming Access
Federation Transparency	True Open Distributed Processing Hallmark - Enables Large-Scale Interworking

Figure 8 provides an introduction to the predominant classes of service interfaces for applications invoking communications services. The first class are interfaces that give access to essential transport services, including establishing, using, and closing communications sessions. These APIs provide application access to the top layer of the OSI-RM layer 7 as well as access directly to layer 4 transport services. Many peer-to-peer protocols fall into this class of interface.

The second class of interface hides the details of network connection establishment from the applications. Remote Procedure Call (RPC) is a well-known example of this class of interface. RPC are used extensively in distributed computing environments, such as OSF's DCE. RPC interfaces are modeled after subroutine calls in procedural programming approaches. A call is made to a remote process and arguments are passed. The RPC, with associated services, handles all inter-communication session management. Although this class of interface provides transport independence, the structural nature of RPC's make object communications difficult.

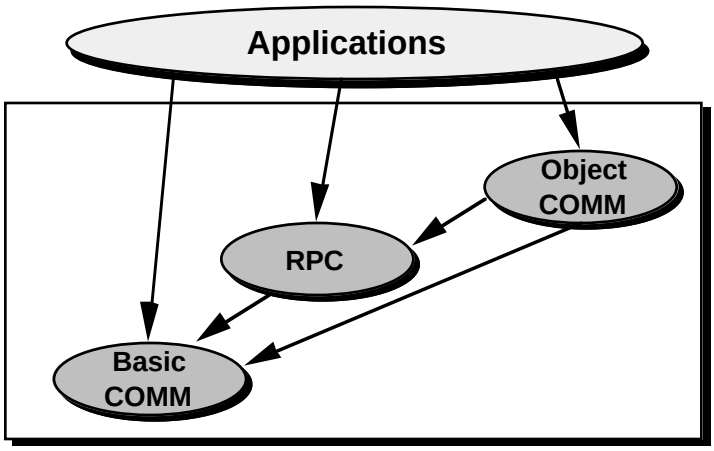


Figure 8. API Maturation for Communication Services

The third class of interface hides the underlying transport, and supports object-oriented application software message passing. This highly abstract interface provides transport independence and support for advanced distributed and collaborative computing concepts that will be required to help meet the technology drivers discussed in section 3.1.

The best way to protect against unwanted interface modifications during the ECS Lifecycle is to select as high-level and abstract an interface as possible, then integrate the interface to the specific underlying communication services, as required. This approach is followed in the communications sub-architecture. At the same time, support for legacy services through classic communications service interfaces is permitted.

3.2.2 Open Distributed Processing (ODP)

The technology drivers of section 3.1 require the selection of the ISO ODP framework for the logical architecture of the ECS communications sub-architecture. The Descriptive Model (part 2) of the ODP reference model (RM-ODP) aims to provide transparent sharing of services (and

resources) over different implementation architectures, different networks, and different operating systems. The objective of the Open Distributed Processing (ODP) reference model is to enable distributed system components to interwork seamlessly, despite heterogeneity in equipment, operating systems, networks, languages, database models, and management authorities. An ODP system provides a set of mechanisms which mask underlying heterogeneity from users and applications. The ODP mechanisms address a set of transparency properties which are fundamental to seamless interworking. Currently, ODP is at the ISO committee draft level (CD 10746).

ODP utilizes object technology to meet its goals. Objects in ODP embody the ideas of modularity and data abstraction that will be necessary to meet the technology drivers developed from scientist interviews. The essential properties of service distribution - separation, isolation and autonomy are captured by the utilization of objects. ODP is based on avoiding assumptions that would otherwise inhibit a true distributed architecture. These assumptions to avoid include the assumptions of central control, single global name space, global shared memory, global consistency, sequential program execution, total failure, locality of interaction, fixed application locations, direct client/server bindings, and system homogeneity.

ODP details five fundamental viewpoints for information system modeling. A viewpoint is an abstraction mechanism which allows a perception of a system emphasizing a particular concern, while ignoring other characteristics that are temporarily irrelevant. ODP recognizes that viewpoints do not comprise a layered architecture. There is no inherent ordering among the viewpoints and there is no implied methodological sequencing. Viewpoints are qualitatively different from one another. The five viewpoints and examples of their use that are planned for application to the ECS-CSMS design are summarized below:

- **Enterprise Viewpoint:**

  Concerned with the social, managerial, financial, and legal policy issues which constrain the human and machine roles that comprise a distributed system and its environment.

  This includes basic requirements, policies, objectives, and user roles.

- **Information Viewpoint:**

  Concentrates on information modeling, flow and structure, and information manipulation constraints of the distributed information system.

  This includes groupings, relationships, flows, and partitions.

- **Computational Viewpoint:**

  Focuses on the structure of application components and the exchange of data and control among them.

  This includes service models and programming functions in the application domain. These include the major models for FOS, SDPS, and the System Management application space.

- **Engineering Viewpoint:**

  Concerns the mechanisms that provide distribution transparencies to application components.

These are the 'psuedo-objects' to the application domain, including protocols, and common facilities and transparency services.

- **Technology Viewpoint:**

  Focuses on constraints imposed by technology and the realized components from which the distributed system is constructed.

  This includes the actual products, devices, and circuits, etc. used to build the system.

The two major viewpoints of immediate concern to the CSMS architecture include engineering and enterprise viewpoints. The engineering model can be described as classes of objects that collaborate together to effect a communications 'channel' for intercommunication. The key object classes include basic engineering objects, stub objects, binder objects, manager objects, nucleus objects, supporting objects, and interceptor objects.

The RM-ODP transparency properties include access, location, migration, concurrency, federation, liveness, resource, failure, and replication transparency properties. These properties can be (loosely) thought of as types of the dominant engineering objects listed above. Further discussion of the engineering model, and transparency aspects of the ODP engineering model, are provided in Appendix A; detailed discussion in available in referenced documentation in the bibliography.

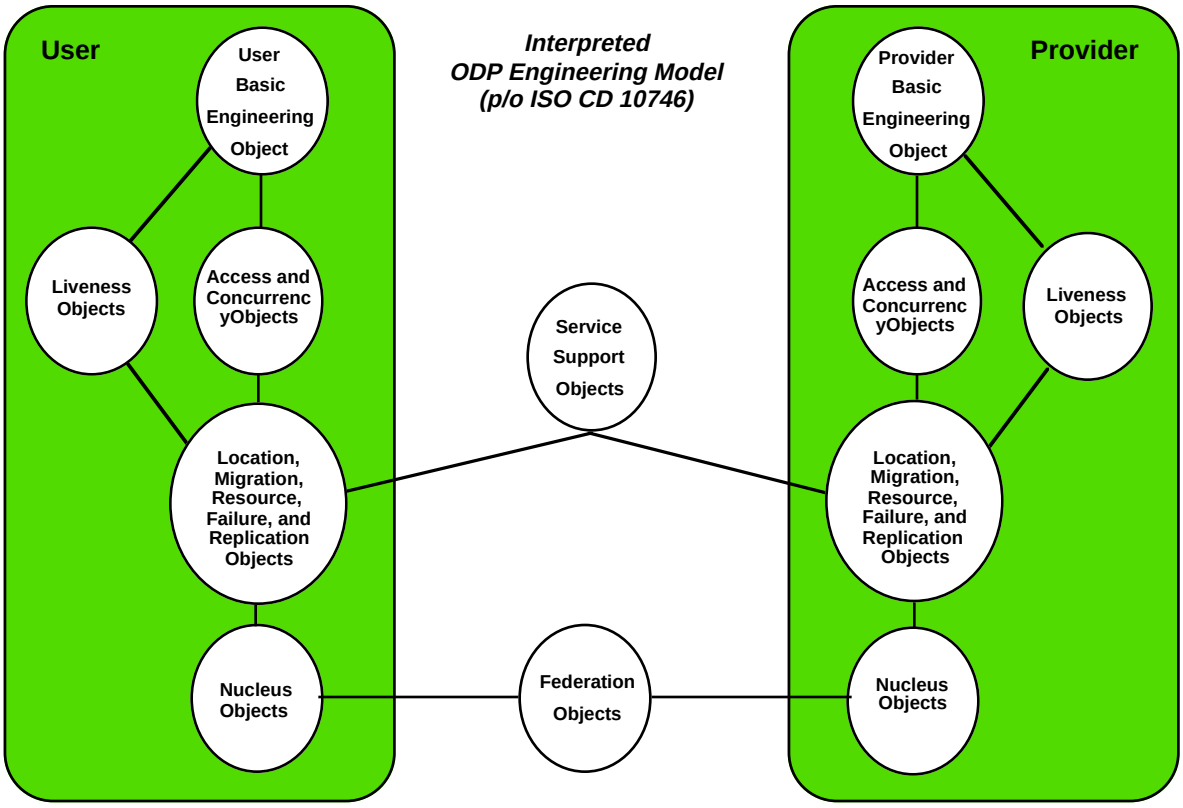


Figure 9. ODP Channel Negotiation with Transparency Objects

As previously stated, ODP aims to provide transparent sharing of services (and resources) over different implementation architectures, different networks, and different operating systems. To share these services, service requestors need to be aware of potential providers of the required services, and need to be capable of accessing the services. In a system such as ECS where user and provider sites are expected to be changing and growing over time, applications must be able to select appropriate servers to meet their needs at run time. This process is known as late binding and is provided by a function called the trader within ODP. The framework is summarized below, while the implementation architecture of section 3.3 develops the trader function within the OMG object model.

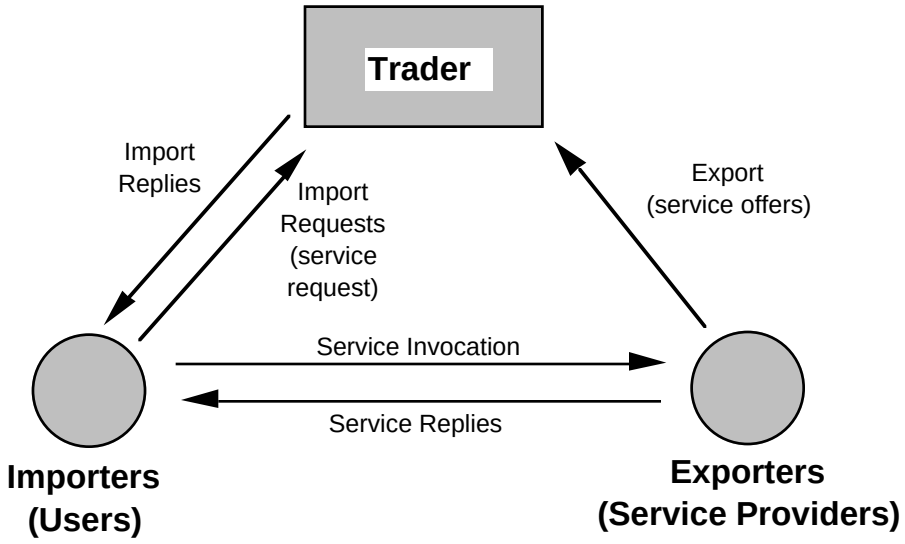


Figure 10. ODP Trader

Figure 10 illustrates the open distributed system trader concept to enable the linking of clients and servers in a distributed system. The service invocation process occurs as follows:

- A trader accepts service offers from exporters (providers) of services when a server wishes to advertise a service offer. A service offer contains the characteristics of a service that an exporter is willing to provide. Service offers are stored by the trader in a centralized or distributed database.
- A trader accepts a service request from importers (users) of services when a client requires a service. A service request is an expression of service requirements made by an importer when a service is needed.
- A trader searches its service offer database to match the importer's service request. And, if required, a trader can select the most appropriate service offer (if one exists) that satisfies the importer's service request. The matched list of service offers or the selected service offer is returned to the importer.
- After a successful match, the client can interact directly with the matched service.

The matching and selection of the appropriate service at run time by a trader allows client objects to be configured into an open distributed system without prior knowledge of server objects that can satisfy their requirement. Note that the same object can be an exporter of one service and an importer of another service. In fact, an importer or an exporter can be another trader. An importer and its corresponding exporter can be co-located, physically removed from one another, or dynamically relocatable. The distribution transparency function of an open distributed system hides the location differences.

Services are defined by ODP as functions provided by objects at computational interfaces. As such, traders can be used to perform all kinds of service negotiations, from real-time trading to large-scale trading. By defining service types and trader types to match the various service types, negotiation by the trader for channel establishment between users and providers is enabled in heterogeneous environments.

3.2.3 Distributed Object Management Systems

The computer industry has been moving toward the establishment of a common application interface standard that can greatly simplify the development and integration of advanced distributed applications. In such an environment, an intermediary, acting on behalf on the requesting client, handles all activities associated with establishment of the entire communication channel between the user/client and provider/server. Objects (clients)who know of a service's existence request services from other objects by issuing a request for service to an intermediate object using only an object reference and the service to be performed. If the service is not initially known by the client, then the client request is actually a request for service, and the intermediary would provide *that* service to the client, acting as a trader in fulfillment of a service import.

The distributed object management systems (DOMS) architecture provides all services required to accept requests from clients, negotiate the service request based on available services and policy, deliver the request to the correct object (if it exists), and return a response indicating the result or outcome of the request. The client is unaware of the underlying complexity of handling the service request, allowing maximum application modularity and interoperability. The DOMS may enlist the participation of one or more other objects before completing the service request without the requesting client knowing about it.

A DOMS consists of an arbitrary number of distributed (physical) nodes, and clients. Each node supports one or more application programs, database systems, and objects. Together, the nodes constitute the system's computing resources. DOMS clients request operations to be performed by the resources. One (or more) distributed object managers act as intermediaries between clients and resources. Distributed object managers make the system's computing resources appear as objects (whether they are object oriented or not). They allow clients to make requests involving resources that reside anywhere in the system. Clients do not need to know the location and implementation details of the resources. Clients and resources connect to the distributed object manager through software interfaces that translate requests and results passing through them to the forms required by the various components.

The DOMS conceptual architecture is depicted in Figure 11. An example of the flow between services that may conspire to fulfill a client request, based on a mix of static and dynamic

invocations, is shown in Figure 12. Components of this example have been implemented by, among others, MITRE DISCUS using Object Request Broker technology from the Object Management Group (OMG). Regardless of the underlying service complexity in fulfilling the client request, minimal interface to the DOMS is required by the client, with no knowledge of the communications and internetworking environment.

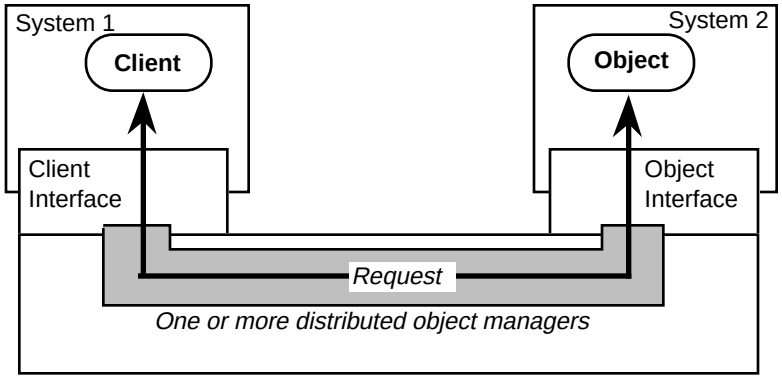


Figure 11. DOM Architectural Concept

The scenario below is a very hypothetical and greatly simplified scenario where a client makes a query for data. This scenario is an extreme case: it is usually the case that services are well-known, with agreed sets of properties and values. Additionally, the implementation of such a scenario within ECS would be toward the concrete and deterministric parts of the scenario first, with the more 'visionary' parts over time as time and money permit. The scenario reflects the potentially achievable state of distributed computing in the next 10 years. The services are actual OMG object services that will be discussed in detail in section 3.3. In this scenario, the following actions might occur:

1. The client initiates a query for certain instrument data. It is unknown by the client if the data exists, or if it needs to be created. Information about the client's local environment and user demographics (user class, context) may be passed with the query. The query requests the data delivered in a specific format.

2. The DOM determines that request must first go to a trader (or service negotiator) service to analyze how best to direct the query to an appropriate query service. In a smaller system, context and local environment parameters might be implicitly assumed, negating the need for a trader.

3. The query is passed to a trader service of type query. This trader might contain detailed information about available query services that could understand the user demographics (such as context of query), and local environment constraints (such as query language and communication protocol support).

4. The trader of type query does initial analysis of the query to determine the existence and reference of a query service that could fulfill the client's request. A full service match is made.

5. The service reference and the query are passed back to the DOM, or another instance of the DOM.

6. The DOM directs the query to the appropriate query service.

7. The query is transferred to the query service.

8. The query service, appropriately selected for the user's demographics and local environment constraints, performs an analysis of the query. As required, the query is parsed, and additional accesses to the trader (or a local instance of it) are made to identify sub-query service references. Assuming no parsing is required, the query service determines (most likely through a naming service, not shown), the object reference of the requested data. The query service additionally determines that the data request requires the join of two existing data sets, but in a different data format than presently available. Unfortunately, the existence of the required data interchange service is unknown.

9. The query instructions are returned to the DOM, or another instance of the DOM.

10. The DOM directs the query instructions to another trader, this one of type data interchange.

11. The query instructions are sent on to the trader, type data interchange.

12. The trader examines its repositories to find a match for the requested data format. A complete match is not made, only a partial match is made.

13. The trader, through an event service (not shown) alerts the client of the partial service match.

14. The client elects to continue the request and receive the data in the available format.

15. The trader of type data interchange obtains the reference for the appropriate data interchange service(s).

16. The retrieval and conversion instructions, with the service reference, are returned to the DOM, or an alternative instance of the DOM.

17. The DOM directs the retrieval/conversion instructions to the data interchange service.

18. The instructions are transferred to the appropriate data interchange service.

19. The data interchange service retrieves the data (working through potentially archive, persistence, and lifecycle services - not shown), combines, and performs a data conversion of the results.

20a. The data, in its packaged form, is returned to the DOM.

20b. Alternatively, client local environment constraints could have the data flow direct to the client using a preferred protocol mechanism.

21. (Optional) The DOM directs the data to the client.

22. (Optional) The data result is returned to the client.

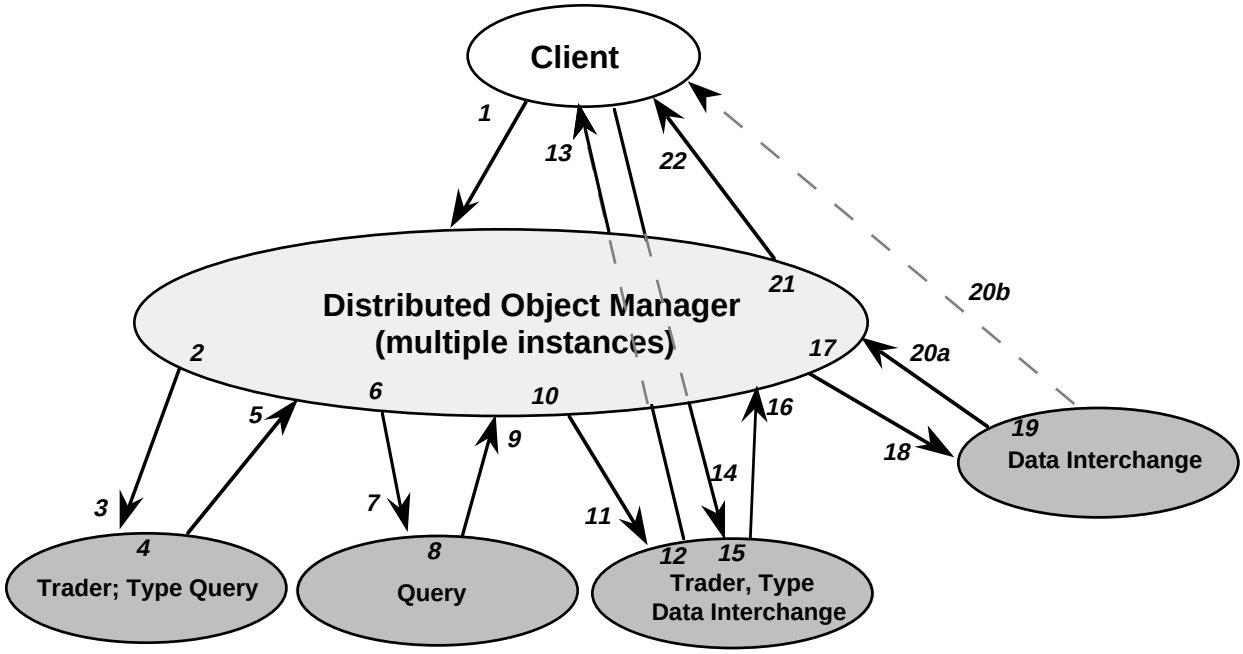


Figure 12. Example Service Flow Scenario using DOMS

Note in this scenario that any of the services may be moved very close to the client to improve performance. Provisioning of the user demographics and local environment constraints (including Quality of Service specifics to performance and availability) drive the implementation (physical design) of the services, and provide a foundation for service negotiation in a heterogeneous environment. The fundamental access to the services, using well defined interfaces, will remain unchanged through variations in client demographic and environment constraints, and multiple service implementations.

3.3 Implementation Architecture

3.3.1 Important Architectures

Early investigation into implementation of specific aspects of the ECS communications sub-architecture included the examination and review of architecture and design details of several systems, technologies, standards and consortia activities. Dominant activities have included a review of X/Open framework activities, discussions with an X/Open fast-track member on Federated Naming (XFN), OMG Technical Committee meeting attendance, MITRE discussions on future OMG object services (Query, Data Interchange, and Trader), examination of NIIT Earth DS and Ellery Open Systems, OSF DCE technical consultancy, Project Pilgram staff discussions, S2K documentation review; and ISO review of ODP, Trader, and QoS-related documents. A summary

of the analysis of primary systems and technologies applicable to the communication sub-architecture are further provided in this section.

Table 2 details a mapping of the driving technologies identified in section 3.1 against primary systems/technologies applicable to ECS. As can be seen from Table 2, the selection of OMG CORBA technology (and OMG Object Services) has the greatest long-term potential to the ECS. DCE with extensions holds good promise as an infrastructural foundation to get to OMG. A summary of the early architecture analysis is provided below - further discussion of DCE and OMG CORBA is deferred to the implementation architecture discussions of sections 3.3.3 through 3.3.5.

3.3.1.1 ECS Version 0 Prototype

The V0 prototype, and its potential reuse to the ECS, is discussed in the V0 Migration Study and is not repeated here. The V0 prototype satisfies many of the initial technology drivers for ECS - and many legacy services of V0 will be enveloped by ECS, especially within the internetworking sub-architecture and sub-services of the communications sub-architecture. Significant enhancements to V0 would be required, however, to support the advanced distributed computing infrastructure services required to meet the science mandates for the communications sub-architecture.

3.3.1.2 Sequoia 2000

Sequoia 2000 (S2K) provides several advanced concepts of potential benefit to ECS. Concepts borrowed for architectural work include:

- the goal for a smooth transition between directory to inventory search to visualization
- assigned bandwidth for isochronous communications in support of collaborative processing
- potential use of a database as a naming server

Promising aspects of work involved with S2K includes in-depth analysis of the POSTGRES database, real-time protocols (RTIP, RTTP), Hollywood work on real-time collaboration, SPIMS, the Earth Scientist data language definitions, and the UCB work associated with OGIS (see Appendix G). This work has potential to both the communications and internetworking sub-architectures. Descriptions of the work are not presented in this paper - the reader is referred to sets of available papers about S2K in the ECS library and via anonymous ftp with uc -berkeley.

Table 2. Technology Driver Mapping to Primary Systems/Technologies

Technology Drivers	V0	DCE	S2K	NIIT EDS	DCE w/Ext.	CORBA
Synch. Interprocessing	√	√	√	√	√	√
Asynch. Messaging	√		√	√	√	√
Static Invocation	√	√	√	√	√	√
Explicit Static Binding	√	√	√	√	√	√
Implicit Static Binding	?	√	√	√	√	√
Directory Service/Scalability	P	√	?	√		√
Naming Service/Scalability	P	F	P	P	F	F
Security Service	P	√	?	√	√	F
Object Technology	?	P	P	P	P	F
Time Synchronization	?	√	?	√	√	F
Multivendor Interoperability		√		√	√	F
O/S Transparency		√		√	√	F
Event Processing/Maturity		P	?	P	P	F
Concurrency		√		√	√	√
Internationalized Security		F			F	F
Multiple Language Support		F	?		P	F
Legacy Server Integration				√	√	F
Dynamic Invocation			√	√	√	√
Dynamic Load Balancing			P		√	
Request Brokering				P	√	√
Server Advertising/Scaling			P	P	P	F
Real Time Collaboration			√	P	P	F
Trading						?
Federation Transparency						?

√ = Compliance, P = Partial Compliance, F = Future Compliance, ? = Incomplete information

3.3.1.3 NIIT Earth Data Systems

The NIIT Earth Data Systems application is based on DCE technology using Ellery Open Systems product for legacy server integration and 'dynamic' invocation. The NIIT EDS shows the potential of advanced distributed processing, but does not scale well to a system the size of ECS. Additionally, the Ellery product is limited in operation to a UNIX paradigm and is not currently based on advanced object technology. Although service 'discovery' is enabled via the Ellery toolkit, the architecture is not truly dynamic in that the client must still understand the correct interface characteristics of the service it is trying to invoke, creating a monolithic design of client/server interaction. Still, the basic NIIT EDS architecture provides several insightful lessons learned with regard to legacy server integration and dynamic invocation concepts.

3.3.1.4 OSF Distributed Computing Environment

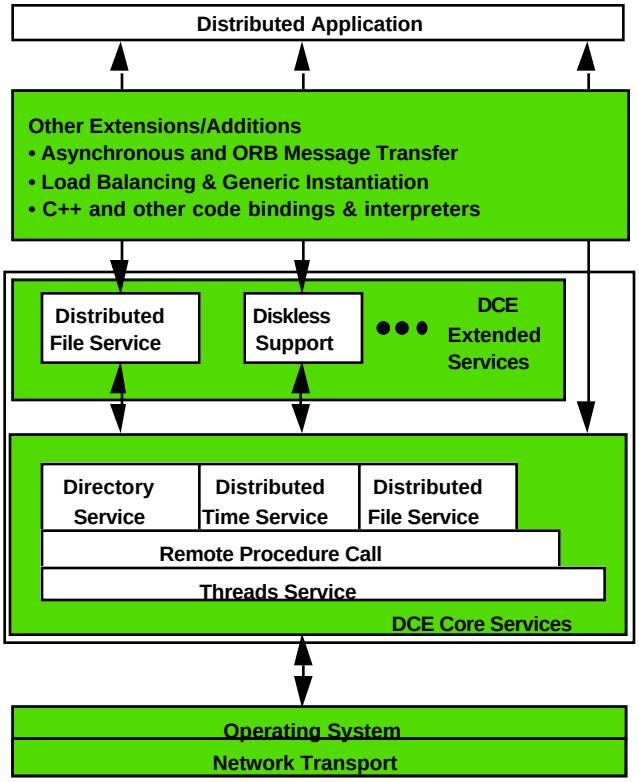
The OSF Distributed Computing Environment (DCE) provides a solution to most of the key technology drivers identified in section 3.1. The core DCE services include threads, remote procedure call (RPC), directory service, security service, and time service. Extended DCE services include distributed file service and diskless support service. DCE is based on three distributed computing models: client/server, remote procedure call, and data sharing. The DCE architecture has been implemented and is available through multiple vendors. It provides an interoperability solution for transport and operating system transparency, and has high immediate benefit to the ECS communications sub-architecture. More information on the DCE Core and Extended services, and the distributed computing models can be found in Appendix B of this document.

3.3.1.5 Project Pilgrim

Project Pilgram is a DCE-experience project out of the Department of Computer Science at University of Massachusetts, Amherst. Early Project Pilgram work included a proof-of-concept experiment of DCE technology. Results of this proof of concepts were very encouraging. Multi-vendor cell integrations included Ultrix MIPS, OSF/1 Alpha AXP, HP-UX, AIX, Gradient Windows/DOS i486 was performed successfully, proving the OSF concept of interoperability. Lessons learned from Project Pilgram include the following:

1. Heterogeneous client/server pairings worked without major problems
2. RPC service time increases linearly with data transfer size (for simple arrays).
3. High throughput rates and network bandwidth utilization is possible.
4. Threads can reduce effective RPC service time.

Project Pilgram has other advanced applications of distributed processing built on DCE which have potential to ECS. These include load balancing mechanisms, asynchronous invocation tools, mail on DCE, and DCE to non-DCE interface experience. Many other COTS and GOTS applications exist as well that can be added to extend DCE functionality, including OSF DME distributed services.



• DCE Based on ANSA Predecessor to ODP (EEC Project ESPRIT)
• DCE-Based CORBA Implementations are predominant

Figure 13. DCE Conceptual Model

3.3.1.6 Distributed Management Environment

DME was examined for the communications sub-architecture for its potential contributions to distributed services, and the access methods to OMG object services within the DME Management Framework. A discussion of the DME architecture is provided in Appendix G. Key DME distributed services of potential value to an advanced distributed environment such as ECS include software licensing, software distribution, and event notification services. The former two are beneficial to the systems management sub-architecture, while the event notification service could potentially provide an implementation of the OMG event service. The DME distributed services will be incorporated within DCE as part of release 1.1. Further investigation into the DME architecture has revealed that software licensing and distribution services may not be the X/Open branded 'standard', and event services are not easily built into the OMG-accepted event service. The software management issues are further pursued in the systems management sub-architecture discussion. The DME event model will not suffice for an OMG event service due to fundamental differerences in the OMG-accepted event model vice the DME approach.

DME Network Management Option includes the X/Open branded use of XMP. XMP is a interface to both SNMP and CMIP/CMIS management protocol services, although the invoking application must specify the underlying management protocol to access or collect information from the managed objects.

The DME (Object) Management Framework, currently on hold within OSF, is closely related to OMG's CORBA in terms of architecture, interfaces, object model , and object services. The Minimum set of OMG object services needed to build distributed applications and services includes:

- Lifecycle, in order to consistently manage creating, moving, copying and deleting objects
- Persistence, to provide storage for an object's private attributes, and to be able to support objects whose lifetime is much longer than that of a single process
- Security, to simplify and automate the authorization of requests on an object
- Naming, to obtain an object reference, given in a readable format
- Events, to support alert communications between objects and administrators
- Synchronization, to support & maintain consistency across multiple, concurrent operations on objects
- Associations and Relationships, to define a common foundation for higher level services that describe complex object relationships, like network maps
- Time, to provide consistent view of time in a distributed system

Additional discussion of DME as it applies to the Systems Management sub-architecture can be found in section 5.3 of this paper.

3.3.1.7 OMG CORBA and Object Services

The open architecture of DCE provides an excellent infrastructural foundation in which to base interoperable CORBA implementations, due to an existing multi-vendor interoperable transport with integrated security features. Many implementations or planned implementations of CORBA on DCE exist today as COTS (HP, Digital, IBM). OSF has recently submitted a specification to the OMG request for specification (RFS) for interoperable ORBs using DCE as the underlying middleware infrastructure. This submission is in competition with six other OMG submissions, however, and will likely require change if it is selected as the ultimate base for interoperable CORBA implementations. CORBA and OMG object services are further discussed in this document as follows: an architecture overview of OMG is provided in section 3.3.2, with a detailed discussion of OMG CORBA and Object Services as it applies to the implementation architecture in section 3.3.3, and reference documents in the bibliography.

3.3.2 OMG Architecture Overview

3.3.2.1 Background

The Object Management Group (OMG) is an international organization of approximately 350 vendors,developers and users. The organization was founded in May 1989 and began operations as a non-profit corporation in October 1989. The organization's charter includes the establishment of specifications to provide a common framework for object-oriented application development. Conformance to the specifications makes possible application development across heterogeneous platforms and operating systems.

In September 1991 the OMG adopted a specification for an Object Request Broker (ORB). The ORB specification defines the communications heart of an object-oriented software framework. This key component enables objects to transparently make and receive requests over a network.

The central mission of OMG is to establish an architecture and set of specifications based on COTS technology to enable distributed integrated applications. OMG's primary goals are the reuseability, portability, and interoperability of object-based software components in distributed heterogeneous environments.

3.3.2.2 Relationship to Other Activities and OMG/ODP Liason

The OMG is one of many consortia and standards development organizations (SDOs) involved in object-based standards development. Figure 14 provides a object paradigm view of how each group involved contributes. OMG has received a Category C liason (OMG document 93-8-7) status with ODP. The agreement enables interaction and formal document exchange between ISO/IEC JTC1 and OMG. The liason does not allow voting privileges, and is particularly focused in liason coordination of the ODP trader.

The OMG has set up a Liason Subcommittee to work on these issues and develop liason with other standards bodies. The other organizations the OMG Liason Subcommittee is involved include ISO/IEC JTC1, ANSI X3, ECMA (TC33 - PCTE), Open Software Foundation, Network Management Forum, North American PCTE Initiative, and X/Open Company. The ISO/IEC JTC1

liason includes SC21/WG1 (Conformance), SC21/WG3 (IRDS), SC21/WG7 (ODP), and SC24/WG6 (PREMO). ANSI X3 involvement is with H2 (SQL), H3 (Computer Graphics and Image Processing), H4 (IRDS), H7 (Object Model), T3 (ODP), T6 (Fault Tolerance), J16 (C++), and J2 (Smalltalk). The X/Open Company liasons include the CAD Framework Initiative, Interactive Multimedia Association, (IMA), X/Consortium, PDES (US Product Data Organization), ISO STEP (TC184/SC4), POSC, ODMG, SPIRIT, and PSWG.

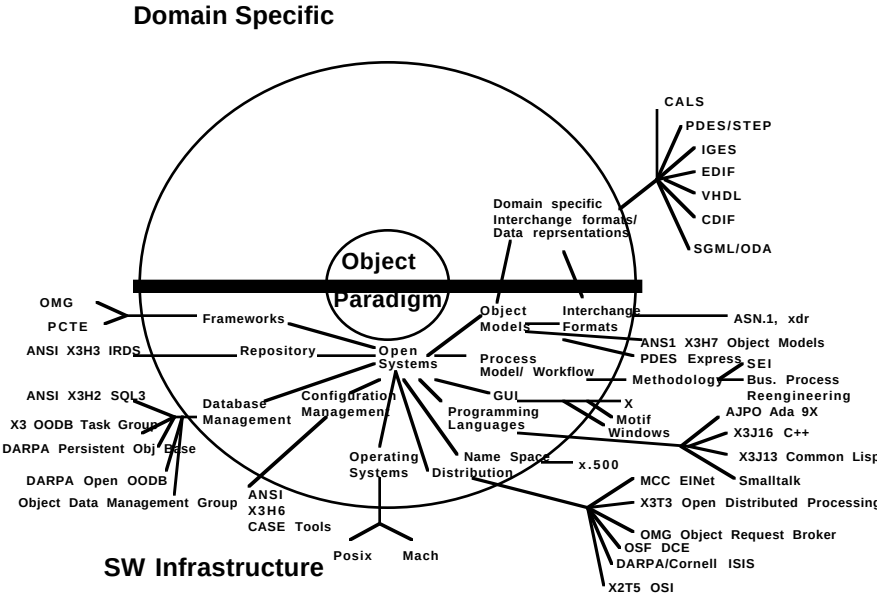


Figure 14. Object-Based SDO and Consortia Activities

3.3.2.3 High-Level OMG Architectural Framework

Through a series of Requests for Information (RFIs) and Request for Proposals (RFPs), OMG is populating the Object Management Architecture (OMA) Reference Model with detailed specifications for each component and service. Appendix C provides a detailed discussion of the OMG architecture. Figure 15 illustrates the high-level architectural object management framework.

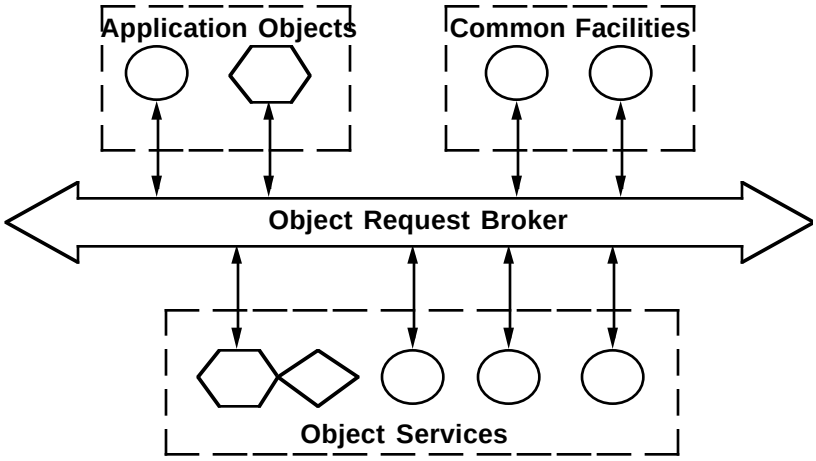


Figure 15. Object Management Architecture

An *Object Request Broker* (ORB) provides the communication backbone for transparently making requests to and receiving responses from objects locally or remotely without the client needing to be aware of the mechanisms used to communicate with, activate, or store the objects. *Application objects* are proprietary or domain specific applications which are not general purpose or reusable. *Common facilities* are general purpose but often domain-specific and typically provide direct functionality to the end-user. For example services which specifically support the construction of network management applications or word processing applications. *Object Services* are general purpose and always domain independent and are necessary to construct any distributed application.

3.3.2.3.1 Object Request Broker

An ORB provides the basic mechanism for transparently making requests to and receiving responses from objects located locally or remotely without the client needing to be aware of the mechanisms used to represent, communicate with, and activate or store objects. As such, it forms the foundation for building applications constructed from distributed objects and for interoperability between applications. Figure 16 illustrates the concept of the object request broker.

The Object Request Broker (ORB) is a core set of object services that enables distributed processing of objects. The core services within the ORB include an interface definition language (IDL) compiler, an interface repository, an implementation repository, a static invocation interface, and a dynamic invocation interface. Clients may access the ORB either statically or dynamically. Figure 17 depicts the ORB and interface types.

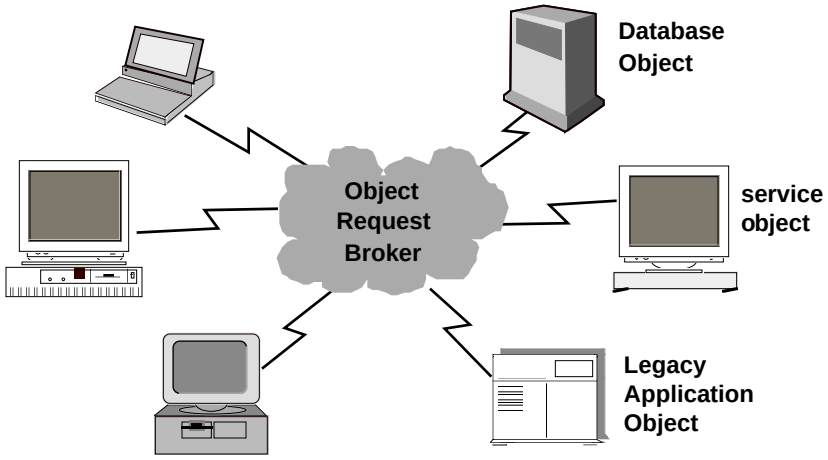


Figure 16. ORB Concept

Structure of CORBA Interfaces

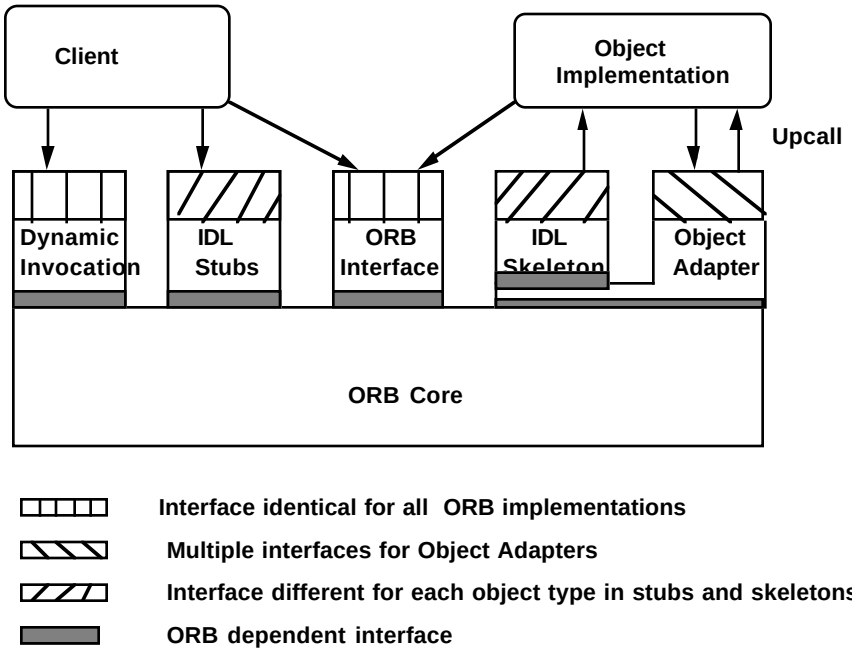


Figure 17. ORB Core and CORBA Interfaces

The ORB provides distribution transparency, in that requests to and responses from an object are made in the same way and have the same semantics whether the client and service provider are located in the same address space, or on two machines on separate continents connected via a wide-area network. In order to do this, the ORB provides access transparency (the source code to perform invocations has the same syntax and semantics whether the target object is local or remote), location transparency (interactions with remote objects behave in the same way regardless of their location) and implementation transparency (interactions are independent of the programming details and data representation of the objects concerned). Objects made available through an ORB publish their interfaces using the Interface Definition Language (IDL) as defined in the CORBA specification.

3.3.2.3.2 Object Services

Object Services are a collection of services (interfaces and objects) that support basic functions for using and implementing objects. Operations provided by Object Services are expected to serve as building blocks for OMG Common Facilities and Application Objects.

An ORB doesn't provide interoperability by itself. Semantic support is provided by Object Services in the form of additional interfaces, protocols and policies. The services are general purpose and necessary to construct any distributed application. Object Services are lower level than Common Facilities and augment the functionality of the ORB. Further discussion of current proposed object services are found in section 3.3.3.1.

3.3.2.3.3 Application Objects

The Applications Objects component of the OMA represents application objects performing specific tasks for users. An application is often built from a large number of basic object types, with some aspects specific to the application and others possibly from the set of Common Facilities. Applications built on this approach provide improved productivity for the developer and expanded functionality for the end user.

3.3.2.3.4 Common Facilities

The Common Facilities component provides a set of generic application functions that can be configured to specific requirements. Standardization leads to uniformity in generic operations and lets end users modify object configurations instead of configuring individual applications.

3.3.2.3.5 Profiles

The OMG Object Model defines a mechanism called profiles. Profiles are groups of components that combine to serve as a useful set of extensions for particular application environment domain, such as a particular language, or SQL3, or ISO GDMO. Profiles can be technology-based; for example databases or programming languages. Profiles can also be application-based, for example CAD or Finance. Vendors will build products that are compliant with profiles for particular domains. Since components have to be compliant with Core features, compliance with a profile means that a product will be compliant with the Core Object Model.

3.3.2.3.5.1 Core

The OMG Object Model defines a core set of requirements that must be supported in any system that complies with the Object Model standard. This set of required capabilities is called the Core.

The Core is the consensus of the least common denominator of atomic features that the OMG membership feels must be supported by any systems that can call itself 'object technology'. These features are identity, typing, operations, and subtyping/inheritance.

The Core serves as the basis for portability and interoperability of object systems across all technologies and across implementations within technology domains. The Core includes a formal model of types, operations and subtyping. The Object Model defines type = interface + (unspecified) magic token, where magic tokens exist solely to allow objects with the same interface to have different types.

3.3.2.3.5.2 Component

The Object Model also allows for extensions to the Core called components which are not required to be supported by all systems. Components are additional atomic features that may be needed in some application domains, but not in others. Examples are relationships, exception handling and attributes.

3.3.2.3.5.3 Core + Components = Profiles

Profiles are composites of the Core, plus one or more Components that make up a useful object model for a specific domain. These domains can be technology specific (DBMS, GUI, programming language, etc.), or application-specific (Earth science, biology, etc.).

Therefore, a Core + Components = Profile. Only Profiles make up useful object models, and compliance will be measured against a profile. Compliance to just the Core or individual Components that don't make up a Profile is irrelevant.

3.3.2.3.6 Object Adapters

An object implementation can access the main functionality of the ORB through an object adapter. The object adapter is the ORB component which provides object reference, activation, and state related services to an object implementation. Services provided by the ORB through the Object Adapter include:

- generation and interpretation of object references
- method invocation
- security of interactions
- object and implementation activation and deactivation
- mapping object references to implementations
- registration of implementations.

It is not necessary for all Object Adapters to provide the same interface, but every instance of a particular adapter must provide the same interface and service for all ORBs for which it is implemented. Some implementations, for example, an object oriented database may have thousands of objects and may not wish to go through an Object Adapter for every individual call. Using a specialized Object Adapter the process may be specially tuned to allow the proper level of performance. It is expected that there will be a few different Object Adapters with interfaces that are appropriate for specific kinds of objects.

Figure 18 shows the structure and interactions of the Basic Object Adapter. The BOA initiates activity by starting the appropriate server. The implementation initializes itself, then notifies the BOA that it is prepared to handle requests. Between the time the program is started and it indicates it is ready, the BOA will prevent any other requests from being delivered to the server. After that point the BOA, through the skeletons, will make calls on the methods of the implementation.

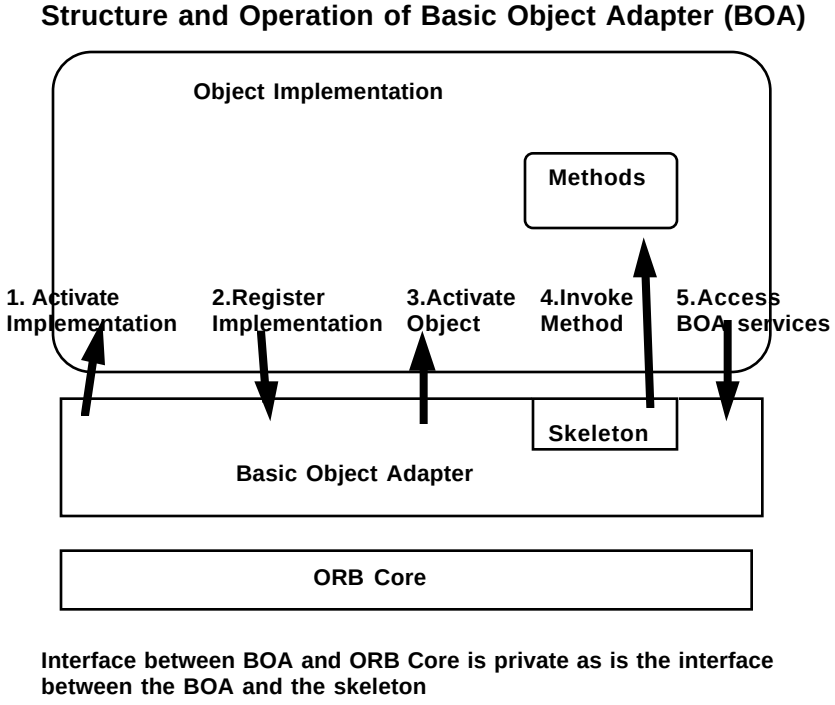


Figure 18. Basic Object Adapter (BOA)

3.3.2.4 OMG/ODP Planning

Several areas of OMG and ODP planning are in process to identify differences and encourage the fast-track development of ODP object services. The major areas of planning include architectural differences in the OMG and ODP architectural models, system management in ODP vs. OMG, and planned CORBA 2.0 extensions to support ODP.

3.3.2.4.1 Engineering and Computational Models

The lack of engineering (viewpoint) objects in OMG is a major architectural difference between RM-ODP and OMG. The locality of objects available to clients at runtime on OMG-based systems are transparent to the client. In RM-ODP this transparency is referred to as the computational viewpoint of a distributed system.

A run-time environment requires the specification of the infrastructure that applications rely on. This infrastructure includes how ORBs communicate, how stubs and adapters work together, or the starting and stopping of processes. In RM-ODP this infrastructure is the engineering viewpoint of a distributed system.

The engineering abstraction is different than the computational abstraction in that the distribution of the objects are no longer transparent, although details concerning the differences between the actual computer hardware remain transparent in the RM-ODP engineering viewpoint.

In RM-ODP, engineering objects are not confused with computational objects and both are generally invisible to each other. The basic engineering object can be mapped to computational object, but a computational object will often map to several basic engineering objects. CORBA 1.1 only recognizes computational objects. The OMG infrastructural objects such as Stubs, Object Adapters, and ORBs are labeled pseudo-objects. The labeling of OMG infrastructure objects as pseudo-objects constitutes the engineering objects in OMG.

3.3.2.4.2 System Management in ODP and OMG

The RM-ODP includes certain aspects of system or object management. The management issues covered include object lifecycle, resource management and policy management. These management functions can be used to build transparencies (e.g., federation, migration, etc.). The RM-ODP includes a life cycle view of object management. The stages in the life of an object include:

- creation - resources are allocated for object template
- service offer - reference to interface provided to trader
- migration - to balance load, reduce latency
- checkpointing - for recovery after host failure
- passivation - when idle to disk, reactivate when invoked
- service offer withdrawel - from trader
- termination - release resources

OMG Object Services architecture provides these management functions in the following way:

- creation - through lifecycle service and factory objects
- service offer - inherent to trading service
- migration - inherent to lifecycle service
- checkpointing - inherent to transaction service

- passivation - programmer defined
- service offer withdrawel - inherent to trader
- termination - inherent to lifecycle service

All RM-ODP objects include a management interface. Objects inherit these supporting mechanisms to reduce programming burden. An object performs a management function by invoking an operation on its management interface. Management objects are not presently envisioned in the OMG framework. Groups involved in these issues include X/Open and the OSF MAN SIG - this is further discussed in section 5.3.

3.3.2.4.3 CORBA 2.0 Extensions

OMG is considering including requirements concerning RM-ODP compliance for the second RFS response of CORBA in the following areas:

- Inclusion of ISO ODP concepts of selective transparencies
- ISO ODP streams as an architectural foundation for multimedia interaction
- Inclusion of ISO ODP node structure and capsule object as generalized Object Adapters
- Inclusion of ISO ODP concepts for the federation of domains
- Inclusion of ISO ODP concepts of transactions
- Inclusion of ISO ODP concepts of replication transparency
- Inclusion of ISO ODP concepts for object clumping
- Inclusion of ISO ODP concepts of trading in support of dynamic reconfiguration

Note that for the potential inclusion of ISO ODP concepts of selective transparencies, users will specify required transparencies in the IDL and the IDL compiler should include code for the required transparencies in the generated stubs. An OMG concurrency transparency (computational viewpoint) may be a quality of service requirement specified in an IDL. The run-time infrastructure must instantiate the appropriate engineering object in the channel configuration to make this possible.

The benefit of this approach would allow a user to pass their local environment constraints to the ORB in order to provide service negotiation via the trader for enhanced interoperability in a heterogeneous environment. As an example, a user might pass local environment protocol stack constraints, so the service requested is made available on the correct underlying communication mechanism.

3.3.2.5 DCE to OMG/ODP evolution concepts

OSF DCE provides an excellent interoperability solution for distributed computing, available today on multiple platforms. The architecture, as it exists today, provides a high degree of transparency support to the platform operating system, and transport levels. The primary evolution concept to OMG involves the early identification and adoption of higher-level OMG-based service interfaces that effectively 'hide' the DCE middleware, allowing a direct migration to CORBA and OMG

object services. The OSF DCE is an excellent foundational platform on which to build interoperable CORBA and OMG Object Services due to its common transport base for multi-vendor platforms, and integrated security services.

OMG is carefully avoiding intensive prescriptive specifications, leaving to the vendor to decide how best to implement the ORB and specific object services. As an example, Hewlett Packard and IBM have co-developed a DOM architecture using DCE services in collaboration with OMG object services. The IBM Distributed System Object Model (DSOM) and HP Distributed Object Management Facility (DOMF) integrated with OSF DCE is shown in Figure 19. The environment contains a CORBA-compliant Interface Repository, enterprise-wide location service, authentication and authorization services, and a set of object services providing persistence, naming, properties, events, and lifecycle management. Services provided by DCE are leveraged, where appropriate, in implementing location and security services. Note that in the DSOM/DOMF architecture, shared memory and sockets are available in addition to DCE RPC. This supports the optimization of object message passing when objects are resident on the same machine or machines of like architecture. The message passing mechanism is made transparent to the application developer and client.

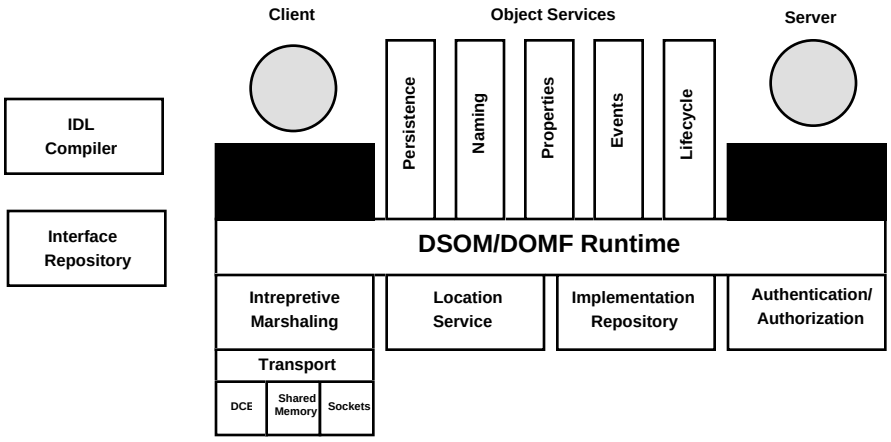


Figure 19. DSOM/DOMF Architecture

OSF has recently submitted to OMG a specification placing DCE as the core for CORBA 2.0 interoperability. The submission includes input from HP, DEC, NEC and HyperDesk. Although IBM is not part of the submission, they may endorse the specification. A separate submission has been placed by IBM using DSOM over an alternative transport infrastructure. HAIS has talked with the OSF team to explore the potential of being I&T site for CORBA interoperability development. The submission to OMG for CORBA interoperability is based on a gateway approach, where messages from one ORB's protocol are translated to another ORB's protocol. This already has been demonstrated by DEC with their OMG ORB to Microsoft OLE ORB gateway. An alternative approach is to force conformance through the homogeneous adoption of a neutral exchange format, where all ORBs intercommunicate using the same protocol. The benefit of the first approach over the second is a recognition of the heterogeneous nature of distributed

processing systems development in the industry. The disadvantage of the first approach over the second is the N-squared potential explosion of gateways required for interoperability.

The selection of OMG-based service interfaces for initial program application design negates the need at a later date to modify code using a DCE infrastructure. This fundamental architectural concept was described in section 3.2.1. In order to protect (non-OO) legacy application investments, special adapters to encapsulate legacy code may require development. Examples of these adapters have already been developed through MITRE and could prove beneficial to ECS as GOTS. Careful and deliberate selection of API sets that map to OMG Object Services, abstracting the underlying DCE services will provide total protection to application developers and effect a smooth transition to CORBA and OMG Object Services. This currently is an area of work with a HAIS OO pilot project.

3.3.3 Communications Sub-Architecture Discussion

3.3.3.1 Heritage Services

The heritage services of the NASA ECS V0 prototype provide significant functionality in OSI-RM layers 5-7 in the areas of electronic mail, file transfer, directory, and information search and retrieval services. The architectural approach in the following discussion keeps the existing services in mind in providing an adaptable framework with support for legacy software and service integration with evolvability to an advanced communications infrastructure. The primary integration strategy is by object encapsulation of the legacy service API. Specific heritage services integration strategy is discussed in section 3.3.3.7 after the OMG architectures are fully explained.

3.3.3.2 OMG Object Services

3.3.3.2.1 Specification Approach

Object services are specified by a set of IDL interface definitions and a description of operation behavior and request sequencing. These interfaces can either be used as types from which application object subtypes can be derived, or services which can be requested by a client via an ORB. In this way, the object services are specified without specific detail to the object implementation. OMG object services can include the definition of multiple interfaces, potentially forming an interface hierarchy. The CORBA IDL supports multiple inheritance, so application objects and service provider objects that participate in or use a service only need to implement those interfaces that are relevant to their role in the service. This additionally supports a non-monolithic approach to service definition: services are amenable to partitioning into separate interfaces and implementation by distinct technologies. The IDL in which the object services are defined can transparently support both library-style and server-style implementations, with support for heterogeneity and programming language independancy. Language maps bind the abstract IDL to a specific language environment. In this way, it is possible to model interfaces in IDL even when the interfaces have a non-object implementation. Language mappings from the IDL to the specific implementation provide a uniform access to both procedural and object-oriented paradigms, further supporting implementation independence of the specifications. The IDL additionally supports interface subtyping. This allows flexibility for object services to start 'small', and incrementally be

extended by adding operations, through interface subtyping, over time. This particular feature has enormous benefit to the evolved introduction of trader services.

In summary, OMG specifications are defined by the interface, and not by the implementation. OMG has acknowledged that there usually is no single 'best' implementation. Multiple implementations are likely to be needed in a heterogeneous environment. Additionally, the object services are general purpose and application domain independent. The object services defined in this section are necessary to construct *any* distributed object application. The common facilities (described in section 3.3.3.3) are more application-domain dependent services that are of specific use to ECS application classes.

3.3.3.2.2 OMG Organization of Object Services

The Object Services Roadmap (OMG TC Document 92.8.5) summarizes OMG's vision for Object Services. The scope and ordering for the phased completion of the Object Services are shown graphically in Figure 20 below.

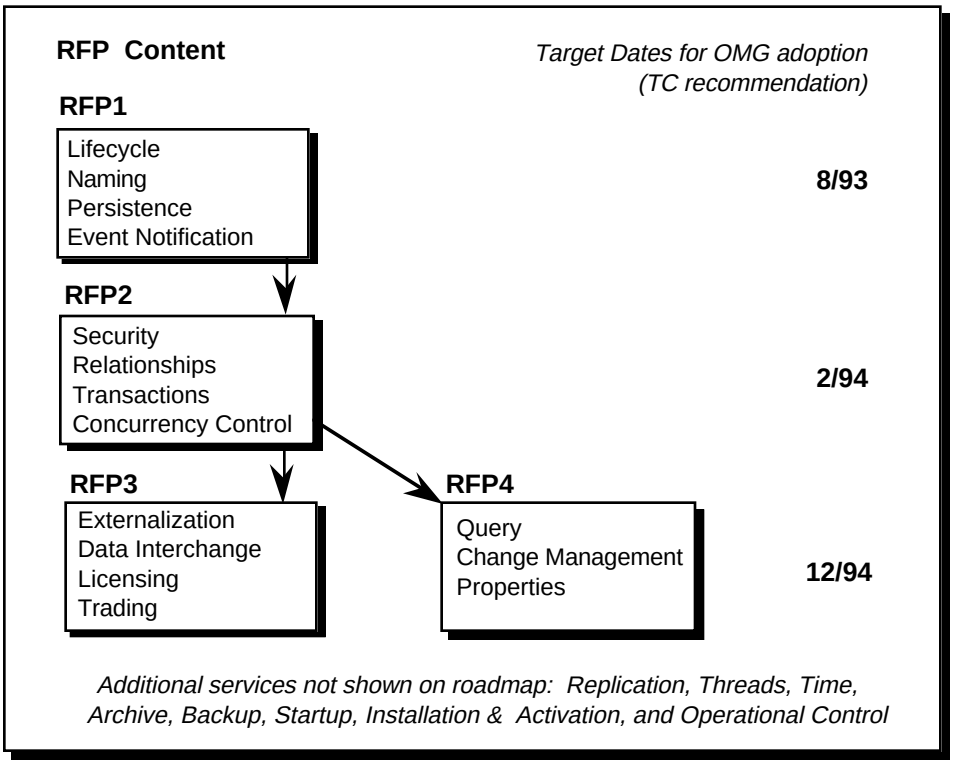


Figure 20. Object Services RFP Timetable

The object service specifications will evolve over time. The service groupings were selected as sets of services largely orthogonal to each other initially, with increasing interdependencies toward the later stages of service solicitation. Section 3.3.3.2.3 identifies the primary set of services recommended by OMG, ordered first by ECS mission critical, essential, and success, then within this initial ordering, by approximate degree of maturation. Services toward the end of each section may never actually get implemented, as the needs for these services will be modified as the higher prioritized services are defined. As an example of this, the original OMG Architecture for object services (August 92) recommended two system management services not found in DME or X/Open literature dated late 1993.

Additional details of the object services can be found in the OMG Object Services Architecture Document, the Joint Object Services Submission (JOSS) Documents, and the Common Object Services Specification, Volume I (COSS). As of November 1993, specifications for event notification, lifecycle, and naming services were approved by OMG. Work was in progress for the persistence service, with two proposals being merged into one (resubmitted to OMG as of March 1, 1994). RFP's were out for relationships, externalization, synchronization (concurrency), and time management services, and a working group was formed for the security service. Detailed phasing of COTS available implementations and implementation strategy of the primary object services by release will be discussed in version two of this document.

3.3.3.2.3 Object Services Classification

3.3.3.2.3.1 Mission Critical Services

3.3.3.2.3.1.1 CORBA

The OMG ORB, although not strictly an object service, falls into the mission critical category. Non-interoperable implementations today, and interoperable implementations are beginning to appear. All services are dependent on the ORB to implement object dispatch. Additionally, all services are dependent on the IDL to express interface specifications, and the OMG Object Model, to describe OMG objects, expressed in IDL. CORBA itself might have dependencies on the interface and implementation repositories to store CORBA interface definitions and implementation objects, respectively. CORBA is critical because it is the basis of which all other services will reside.

3.3.3.2.3.1.2 Interface Repository

The interface repository is a component of CORBA that falls in the mission critical category. The implementation of CORBA includes an implementation of the interface repository. The interface repository supports the management of object interface definitions. Although not required for static service invocations (using stubs and skeletons), the interface repository is a critical part of the ORB required to support dynamically invoked services.

3.3.3.2.3.1.3 Implementation Repository

The implementation repository is a component of CORBA that falls in the mission critical category. The implementation of CORBA includes an implementation of the implementation repository. The

implementation repository supports the management of object implementations. The implementation repository is a critical part of the ORB and required to support other services.

3.3.3.2.3.1.4 Naming Service

The Naming Service provides mappings between names and CORBA object references. It provides the ability to bind a name to an object relative to a naming context. A naming context is an object that contains a set of name bindings in which each name is unique. This service has been recently defined in a submission to OMG as part of COSS.

Several interfaces are defined within the Naming Service. The *NamingContext* interface provides access to operations to bind objects, resolve names, unbind objects, create naming contexts, delete naming contexts, and listing a naming context. The *BindingIterator* interface allows a client to iterate through the bindings using operations that either next binding, or returns at most the requested number of bindings. A destroy operation on the iterator is included. Two additional interfaces are specified which assist in the creation and maintenance of *names library*. The *names library* hides the representation of names from client code, providing efficient creation, manipulation, and transmission of names within application code.

Implementations of the Naming Service can be potentially done by an abstraction of the DCE cell directory service, or some relational database. Additionally, work on Federated Naming within X/Open (XFN) should provide a canonical interface to federated naming service implementations, including DNS and X.500 directories.

3.3.3.2.3.1.5 Event Service

The Event Service allows objects to dynamically register or unregister interest in receiving notification when a specific event occurs. Asynchronous events, event 'fan-in', notification 'fan-out', and reliable event delivery (with appropriate event channel implementations) are supported. The Event Service specification has recently been approved and published by OMG as part of COSS. Implementations of event services exist that may potentially be abstracted into the OMG Event service include COSS vendor implementations, and Project Pilgram work. Event services operate under either a push or a pull model; key interfaces proposed as part of the Event Service includes the *PushConsumer*, *PushSupplier*, *PullConsumer*, and *PullSupplier* interfaces. *EventChannel*, *ConsumerAdmin*, and *SupplierAdmin* interfaces support administration of the event service.

The Event Service decouples the communication between objects. The Event Service, illustrated in Figure 21, defines two roles for objects: the supplier role and the consumer role. Suppliers produce event data and consumers process event data.

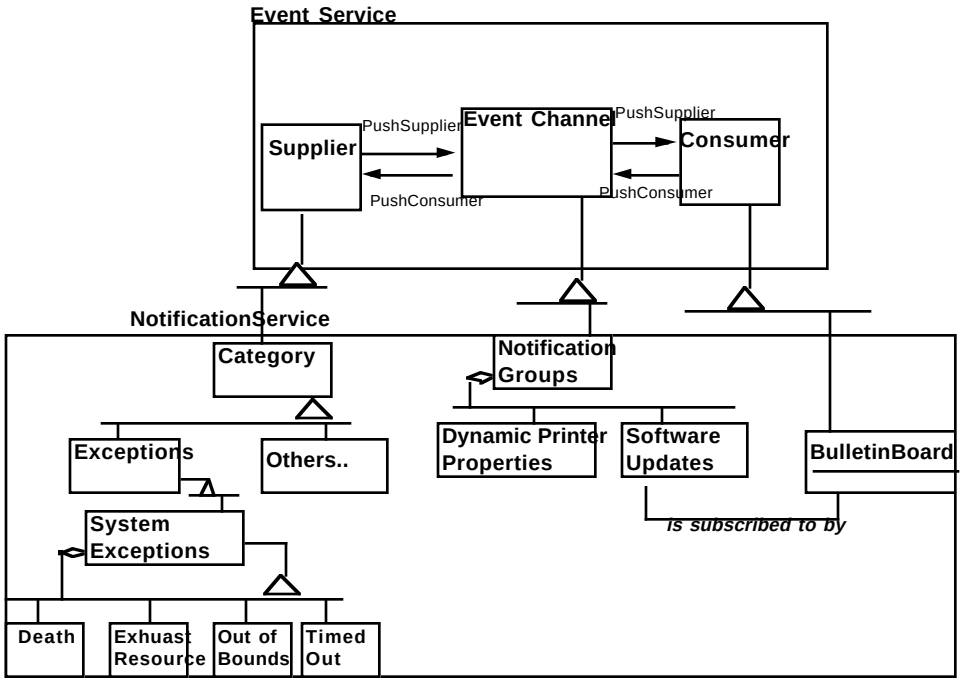


Figure 21. Event Service

There are two basic models for communicating event data: the push model and the pull model. In the push model the supplier initiates the transfer of data to consumers. In the pull model consumers initiate transfer by requesting data from suppliers.

An event channel is an intervening object that allows many to many associations between suppliers and consumers. The event channel also facilitates asynchronous communication. An event channel is both a supplier and a consumer of events.

3.3.3.2.3.1.6 Security Service

The Security service provides access control, encryption, and audit control for objects and interfaces. The DCE security services are an excellent implementation example of the aim of the OMG security service. The Generic Security Standard (GSS) API, when implemented with DCE as part of release 1.1, will provide internationalized security services using a choice of private and public-key encryption.

To provide security to a typical service request in an OMG environment the additional security features must be included. Figure 22 shows where security features are called by object components in a service request.

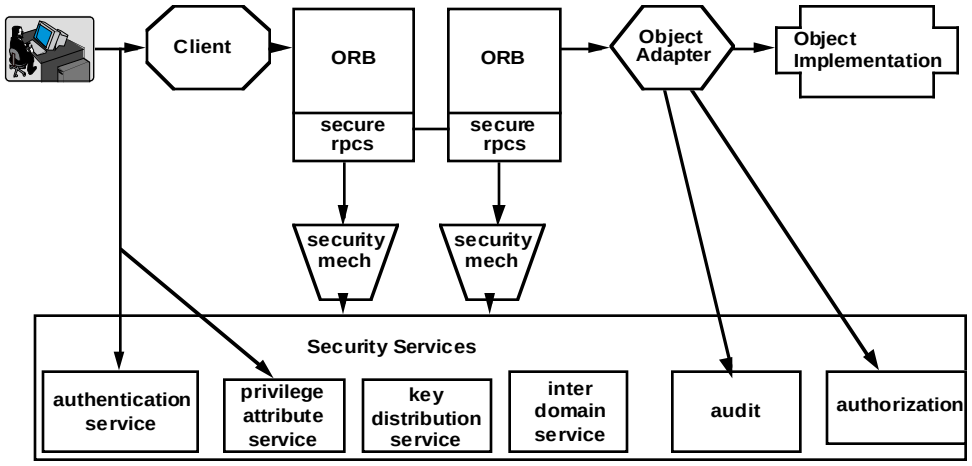


Figure 22. Security Service in a Typical Object Request

A request begins when a user authenticates himself/herself. User authentication is done by calling an authentication service. A privilege attribute service can also called to determine a user's rights. Authentication and privilege selection sets up a user's credentials.

The user calls a client which can invoke an object request. A secure association between the server and client must be established. The request and the user/client's credentials from logon must be securely transmitted to the correct object. At the client, the security interface (GSS-API style) is called to initiate the security context for the request and is supplied a security token. At the server the security interface is called to accept and check the token. Neither the client or the server understand the token. The token includes the user/client's credentials, in the form of a Privilege Attribute Certificate (PAC) and also information about encryption keys used to protect the dialog. The client and server having established mutual trust may now transmit protected data. The transmitted data may be protected using dialogue keys. The ORB/secure rpc calls security interfaces to perform this encryption.

When the request reaches the server system, an access control check is made to see if the the user can use the requested method. The Object Adapter or ORB or object implementation may perform this check. The check uses the privilege attributes of the client's credentials and the access control attributes for the server's method.

The method invocation can be audited. A standard interface for auditing should be used. The object implementation may now be a client to another service on the system. The implementation may do this using the user's credentials or its own.

3.3.3.2.3.2 Mission Essential Services

3.3.3.2.3.2.1 Lifecycle Service

The Lifecycle Service provides operations for managing object creation, deletion, copy and equivalence. Because CORBA-based environments support distributed objects, lifecycle services define services and conventions that allow clients to perform lifecycle operations on objects in different locations. The client's model of creation is defined in terms of factory objects, which are objects to create other objects. This service is approved and published by OMG as part of COSS.

Three interfaces are recommended for this service in COSS: the *LifeCycleObject*, *FactoryFinder*, and *GenericFactory* interfaces. The *LifeCycleObject* interface defines operations to delete an object, to move an object, and to copy an object. The *FactoryFinder* interface supports an operation, which returns a sequence of factories. The *GenericFactory* interface defines a generic creation operation, allowing for the definition of standard creation services.

3.3.3.2.3.2.2 Persistence Service

The Persistence Service provides persistence of an object independent of both the lifetime of the client that accesses the object and of the implementation that realizes the object's methods. Persistent objects are placed in a persistent store such as a file system, database, or repository. It can be used within ECS to assist in the storage of objects.

Persistence Service Architecture

The Persistence Service is an open architecture. It supports a variety of storage services. It is lightweight, simple, powerful, robust, and integratable. Some implementations can be used without using all of its parts. It allows access to non-object storage including relational databases, files systems etc.

A persistent object is an object that exports its persistence behavior. The object supports both a persistence interface as well as a functional interface. For example, a spreadsheet supports its functional interface as well as a persistent interface. There are different ways to create a persistent object. A class can generate a PID. These PIDs cannot change. A PID may be generated by a database when creating object-PID binding. These PIDs might not be changeable. A document might create a PO that allows PID to change during objects' lifetime. Persistent objects are quite flexible. A user can:

- move an object's persistent state to a new DataStore;
- copy or backup an object's persistent state;
- bring an object's internal state back to where it was (restore)
- prevent changes in an object's internal state from propagating to its persistent-state
- get an object's PIDString and give it to someone to
- share its persistent state without sharing object

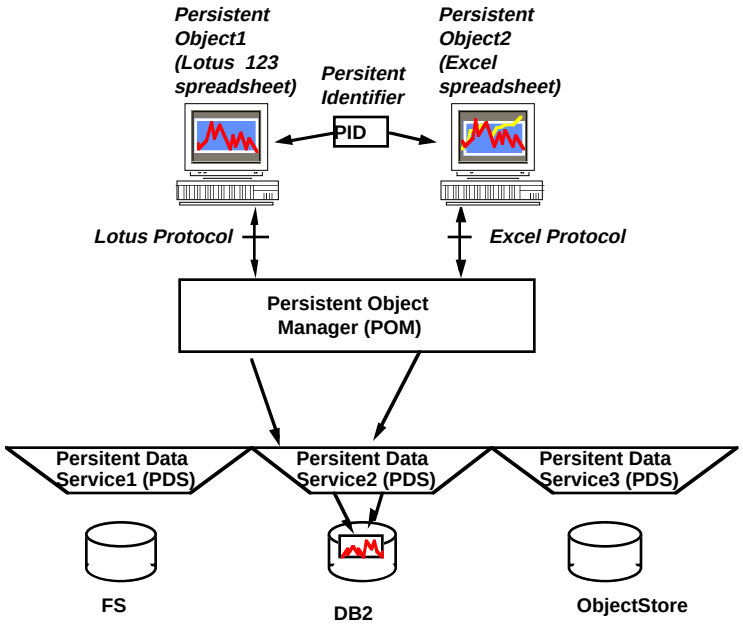
The Persistence service also defines persistent object (po). These objects do not support PO interface. These objects are provided clients do not need extensive control of persistence.

A Persistence Identifier (PID) is a way of specifying a unit of persistent state. The purpose of a PID is to encapsulate the location of object's persistent state. The PID is the only client visibility of where persistent state is stored and the Datastore interface used to access it. The PID supports sharing of an objects persistent state. For example, if two users both work on same spreadsheet, one might like Lotus 123 and the other Excel. If the one user gives the PID the other user can create the object and use the data in their preferred format.

A Protocol is a means of interaction between an object and the persistence mechanism. A Protocol specifies the way to get persistent data in and out of an objects. The actual data transfer details depend on Protocol. Different Protocols are used to support:

- range of data sizes (byte to megabytes/object)
- range of access patterns (all, partial, query)
- range of frequencies (startup/shutdown, incremental updates)
- range of models (single-level store, upcall/downcall)

The Persistent Object Manager connects things together, it interprets PIDs and supports plug-and-play of different implementations. A (POM) provides a uniform interface for the implementation of an object's persistence. Single-level-store native objects do not need the POM and can access the PDS directly. The POM routes persistence operations to correct PDS by using protocol + datastore_type to route to a PDS.



The Persistent Data Service (PDS) is an interface class that supports combinations of Protocol (gets data in/out of object) and datastore_type (gets data in/out of Datastore). The PDS must:

- at least support the operations used by POM
- interact correctly with whatever objects required by the specific protocols it supports
- interact correctly with the specific Datastore it supports

A Datastore is the underlying storage mechanism for persistent state. It may or may not be exposed. Direct access between the persistent object and the Datastore is possible, if supported. Examples include, file system, standard relational database or object oriented database.

3.3.3.2.3.2.3 Relationship Service

The Relationship Service provides for creating, deleting, navigating, and managing relationships between objects. Relationships must support modeling object associations and semantics.

3.3.3.2.3.2.4 Transaction Service

The Transaction Service supports atomic execution of one or more operations, providing the ACID property set of transaction processing systems. The X/Open XTP work as developed by Transarc is a good implementation example of this service. This implementation is available over DCE, but a more robust implementation mapped directly to the OMG IDL is required. SQL3 on RDA is another implementation example that is potentially mappable directly to OMG IDL through profile extensions.

3.3.3.2.3.2.5 Concurrency Service

The Concurrency Service defines how an object mediates simultaneous access by one or more clients such that it and objects it accesses remain consistent and coherent. This service differs from the Threads Service in that threads are for short-term synchronization primitives used to implement concurrency control not limited to transactions. Concurrency control is for objects accessed within concurrently executing transactions. This includes providing operations to mediate concurrent operations, distinguishing between various granularities of concurrency control, and resolving deadlocks. Transaction processing could become a requirement for sensitive transfer and replication of information, including accounting data, security and directory information.

3.3.3.2.3.2.6 Data Interchange Service

The Data Interchange Service enables objects to exchange some or all of their associated state. It is used for bulk data transfer of domain-specific object representations. MITRE has an interesting implementation of Data Interchange on the DISCUS prototype. It is MITRE's intent to place a submission to OMG. ODMG is also doing work in this area (see Appendix J for work with data and database technology relevant to CORBA).

3.3.3.2.3.2.7 Licensing Service

The Licensing Service controls and manages remuneration of suppliers for object services. Licensing schemes might range from perpetual floating licenses to renewable licenses to metered usage, and may vary over the life of the licensed product. The licensing scheme may additionally incorporate the value of the information being accessed and establish related charging schemes between users and suppliers of the information objects. DME Distributed Services has a HP-based licensing services that fulfills most of the goals of the OMG service. It may be possible to use this for the service implementation.

3.3.3.2.3.2.8 Trading Service

The Trader Service enables the linking of clients and servers in a distributed system. This service is the matchmaker between the client and the service for objects. A detailed discussion of the trading concept of service import and exports, and trading extended for federation, is provided in section 3.3.5.

3.3.3.2.3.2.9 Query Service

The Query Service supports operations on sets and collections of objects which may result in sets or collections of objects. Queries are operations on sets or collections of instances that have a predicate-based, declarative specification, and that may result in sets or collections of objects. The emphasis in this service is on fine-grained objects. Appendix J details some work in this area, including MITRE, ODMG, and OGIS.

3.3.3.2.3.2.10 Properties Service

The Property Service allows objects which inherit its interface to associate useful information within its' state, for example, a date or title. The recent Naming Service submission has a *names library* operation that may delete the need for this service.

3.3.3.2.3.2.11 Externalization Service

The Externalization Service provides functions for the transformation of an object into a form suitable for storage on an external media or for transfer between systems.

3.3.3.2.3.2.12 Change Management Service

The Change Management Service supports the identification and consistent evolution of objects including version and configuration management. Change management is used to maintain efficiently the evolution history and composition of complex systems. There are legal, accounting, or accountability requirements that specific Change Management systems must satisfy. There are many related standards and references on this service in work.

3.3.3.2.3.2.13 Replication Service

The Replication Service provides for the explicit replication of objects in a distributed environment and for the management of consistency of replicated copies. Operations of the replication service

include a consistency checker of all replicas, a creation for replicas, updating of changes to replicas, destruction of replicas, and the merging of replicas. This service appears to have a high degree of overlap with the recently submitted lifecycle service capability to perform lifecycle services on graphs of objects.

3.3.3.2.3.2.14 Threads Service

The Threads Service gives programmers the ability to create and manipulate threads. Included are threads management, scheduling priorities of threads, policy priorities of threads, and communications capabilities. DCE Threads are a good example of an implementation of the threads service. Being just above the operating system, the threads service is the only OMG service with no dependency on any other object service.

3.3.3.2.3.2.15 Time Service

The Time Service provides a mechanism for synchronizing clocks on multiple machines, and provide a way to periodically synchronize the clocks. DCE Time, using UTC, is an operating implementation of this service.

3.3.3.2.3.2.16 Archive Service

The Archive Service supports the mapping between active and backup object stores. Objects are copied from an active/persistent store to a backup store and retrieved from the backup store to the active/persistent store.

3.3.3.2.3.2.17 Backup/Restore Service

The Backup/Restore service supports backup and recovery of objects. The service restores the environment to a prior state, usually by making a copy of the environment after significant input or reorganization has occurred. Incremental backup may be used; a copy is then used to recover the system after failure.

3.3.3.2.3.2.18 Startup Service

The Startup Service supports bootstrapping and termination of object services.

3.3.3.2.3.2.19 Installation and Activation Service

The Installation and Activation Service provides mechanisms for distributing, activating, deactivating and relocating managed objects. Managed objects refer to clients of the system management services, which may be application objects, common facilities objects, or other object services. The operations of this service may be transferred to object lifecycle and security services.

3.3.3.2.3.2.20 Operational Control Service

The Operational Control Service provides the mechanisms for controlling the dynamic behavior of managed objects. Managed objects are clients of the system management services.

Note: This service, and the installation and activation service, were originally specified using the ISO system management functions and GDMO as reference standards. Since then reconciliation between GDMO and OMG object models, and between ISO SMFs and OMG Object Services has been in work. Recent CORBA-compliant system management architectures (DME, X/Open) do not list the installation and activation service or the operational control service as service dependencies. The definition of these services, available in the documentation, would suggest a non-orthogonal mapping of these two services to other, higher prioritized services within the OMG object services architecture.

3.3.3.2.3.3 Mission Success Services

The object services are seen to be necessary for any distributed object application. As such, all are categorized as either mission critical or mission essential.

3.3.3.3 Common Facilities

The common facilities are built using the object services defined in section 3.3.3.2, but potentially include support to the legacy services mentioned briefly in section 3.3.3.1. This section provides examples of how the object services might be integrated into an application environment. The communications sub-architecture services presented below are evolving and will continue to be refined toward SDR. DME developed a term called adapter object (NOT to be confused with object adapter), that basically provides the encapsulation of a procedural interface. This term, which is called 'adapter' in this paper to minimize confusion, would be an implementation of an object that is built around the particular legacy service. This provides support for legacy applications, and an integration of the legacy environment into an ODP logical framework. An alternative approach is to create a subroutine library interface of an OO implementation to the legacy service. The legacy services relevent to peer-to-peer and distributed processing, although not directly discussed in this section, will remain essentially intact for ECS usage. These services are additional services to be discussed within the SDS documentation.

The examples described in sections 3.3.3.3.1 through 3.3.3.3.4 illustrate some of the flexibility introduced with OO development over the OMG object services. Specific OMG work in this area is currently in progress. More information of the direction of OMG-defined common facilities (which would not include all ECS-specific domain services, such as legacy services), is provided in section 3.3.3.3.5.

3.3.3.3.1 File Transfer Service

The file transfer service shown in Figure 23 is an example of a multi-object service that can be provided by a trader on a Quality of Service basis. Legacy methods of file transfer including ftp and ftam may be encapsulated into objects. A client that is not particular may receive the entire FileService and allow the user to select the type of service he/she interested in.

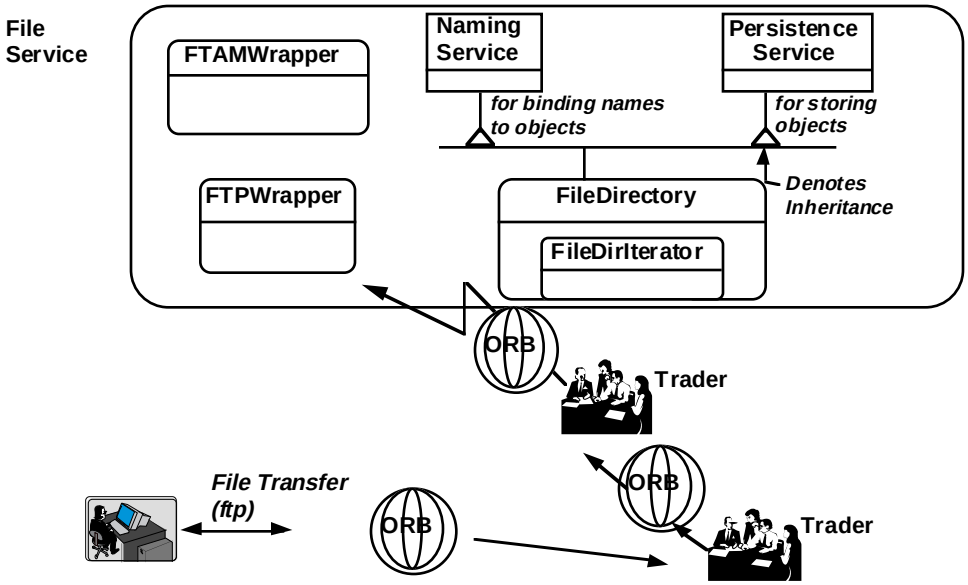


Figure 23. File Transfer Service

Additional analysis is ongoing to investigate specialized object adapters that would improve performance response with strict QoS constraints. The service architecture will continue to decouple low vs high tolerance components of latency. Also, the potential integration of DCE distributed file sharing, and the use of data interchange and query services to build an integrated information search and retrieval service (beyond MOSAIC) can being considered using Figure 23 as a base.

3.3.3.3.2 Electronic Mail

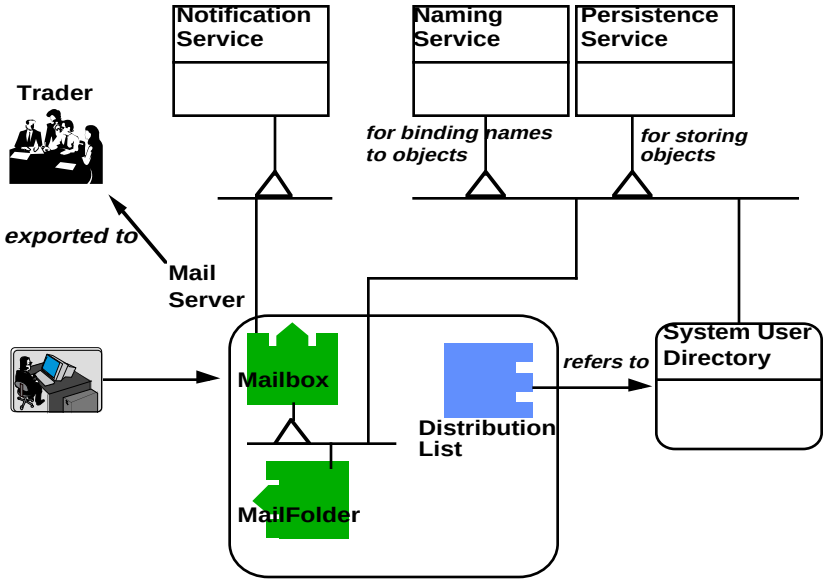


Figure 24. Mail Service

The mail server of Figure 24 is an example of a client and an implementation of different services. The main object in the mail service is the Mailbox object. The Mailbox object would conceptually utilize the bulletin board features of a notification service. A Mailfolder object would augment the facilities of a Mailbox by allowing the messages to be retrieved by name. The Mailfolder object might use services from the Mailbox, Naming and Persistence objects. A DistributionList object would allow a user to save the names and addresses of other users that he/she frequently broadcast messages to. The DistributionList object would conceptually utilize the services of a system user name directory service. The Mail Service could also register its services in part of the system name space by exporting a part of its services with a trader.

The implementation of this is largely available today, using X.400, and SMTP.

3.3.3.3.3 Extensions of Directory/Naming Service

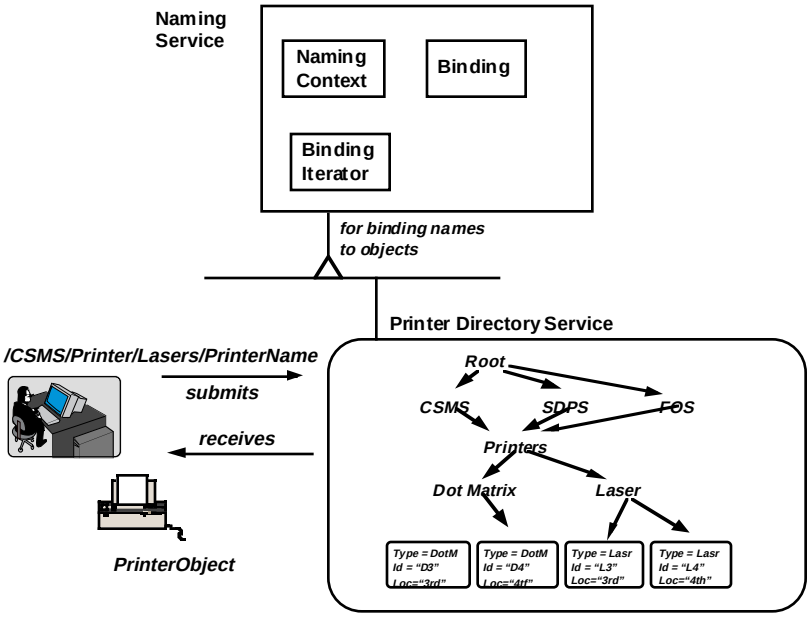


Figure 25. Printer Directory Service

A name binding is an association between a name and an object. A name binding is always defined relative to a naming context. A naming context is an object that contains a set of name bindings in which each name is unique. Different names can be bound to an object in the same or different contexts at the same time.

To resolve a name is to determine the object associated with the name in a given context. To bind a name is to create a name binding in a given context. A name is always resolved relative to a context.

Binding contexts in other contexts create a naming graph - a directed graph with nodes and labeled edges where the nodes are contexts. A compound name is a sequence of names that defines a path in the naming graph to navigate the resolution process. Figure 25 shows an example of a naming graph for a printing directory service.

3.3.3.3.4 Notification Service

Categories in a Notification Service would allow application defined groupings of messages about particular areas or resources within a system, such as printer server dynamic property updates. Categories are derived from Supplier objects.

Notification Group objects, derived from the Event Channel objects, would receive notices, map to a particular Category, and provide notifications to registered consumers.

BulletinBoards are examples of implementations of Consumer objects. A bulletin board object would register (subscribe) with a particular Notification Group to keep abreast of events in decoupled asynchronous fashion.

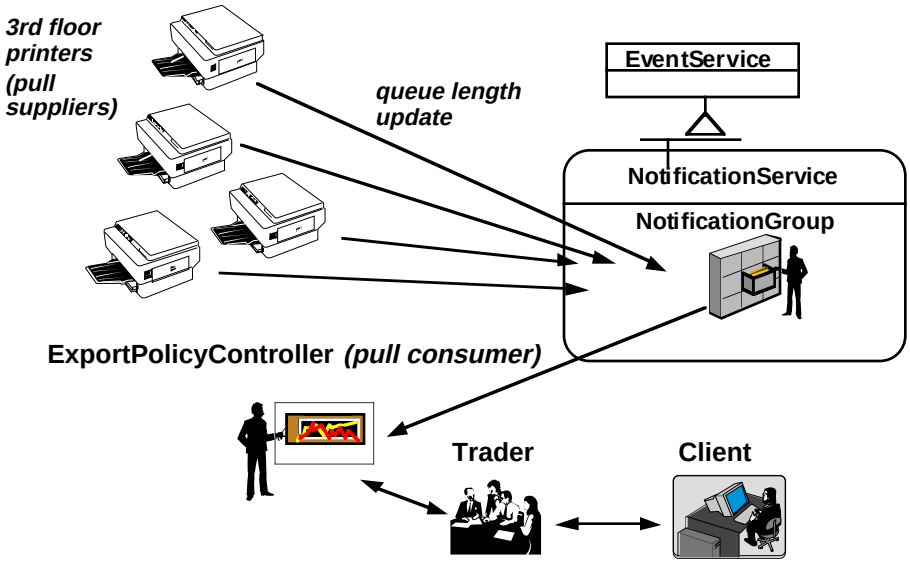


Figure 26. Example of Dynamic Property Notification

Figure 26 shows an example of an implementation of load balancing by an ExportPolicyController using dynamic properties and a Notification Service. After registering with a dynamic printer property notification group as a pull consumer, the client would receive updates on printer queue lengths whenever it requested them. The ExportPolicyController could act as a load balancing agent and recommend the printer that has the shortest queue to the Trader. The Trader could use this information and any selection policy (criteria) information supplied by the client to select an optimal supplier for the client.

3.3.3.3.5 Common Facilities - Summary of Direction

The definition of the Common Facilities for ECS is evolving, embracing technical elements of the user interface, application development, information management, and system management. The ECS project will have many areas where these logical groups of objects will be required. Further analysis is required leading to SDR. At this time, the following are a short list of common facility domains considered pertinent to the ECS project:

- Object Collections
- Object Instantiation Management
- Cataloging and Browsing
- Help Facilities
- Printing and Spooling
- Object Query (Information Search and Retrieval)
- Agents
- Object Interchange
- Clipboard
- Policy Management
- Distributed Object Manager
- Load Balancing

In the near future vendors will offer many varieties of these object collections. All but the last two common facility class listed above (distributed object manager, and load balancing) are areas of common facility development expected to be forthcoming through industry and available as COTS to the ECS project within the next two years. The Component Integration Laboratories (CIL) is an example of an industry-wide association working to adopt and support technologies for the effective integration of software components that provide a COTS base to some of the common facilities. At present, CIL is supporting four technologies:

- OPENDOC, a compound document API
- BENTO, a portable object storage API designed for compound documents
- Open Scripting Architecture (OSA), an automation and scripting API, and
- System Object Model (SOM), an object-oriented, dynamic linking mechanism

These mechanisms will be supported on Windows, Macintosh, OS/2, and UNIX platforms. As these and other organizations offer technology, HAIS will continue COTS vendor negotiations and early prototyping ECS project integration.

Policy management and agents as they apply to distributed systems management will most likely be classified as a management service for system management application domains, which is further discussed in section 5.3.

3.3.3.4 Application Objects

There are no application objects anticipated for the communications sub-architecture. Recall that application objects are proprietary or domain specific applications which are not general purpose or reusable. It is probable that application objects may be built in other parts of ECS from basic object classes, with some aspects specific to the application and from the set of Common Facilities. Applications built on this approach provide improved productivity for the developer and expanded functionality for the end-user. The objects that are shared between applications fall into the common facilities, either developed as part of the communications design, or SDPS design. Some preliminary examples of these common facilities were described in section 3.3.3.2 and are subject to ongoing analysis.

3.3.3.5 Profiles

Recall that profiles are groups of components that combine to serve as a useful set of extensions for particular domains. Appendix J details alternative databases and query languages that may require profile development by their vendors. Beyond databases, ECS will require profiles for alternative programming languages. Profiles for C and C++ are defined: interest in Fortran, Ada and Smalltalk has been expressed. A profile mapping of DCE idl to CORBA IDL is in progress. Specific application profiles may be required to support advanced information search and retrieval services defined within SDPS, such as Z39.50. These areas will require further analysis. The vast majority of industry and government are readily adopting the OMG concept of a Core + Component = Profile, so finding implementors of the Profiles should not be difficult.

3.3.3.6 Object Adapters

Recall the object adapter is the ORB component providing object reference, activation, and state-related services to an object implementation. Some implementations with thousands of objects may require an object adapter specially tuned to allow the proper level of performance. It is anticipated object adapters will be required for the following key interfaces:

- Management interfaces
- Naming and Location interfaces
- Database Query and Data Interchange interfaces
- Trader interfaces
- High performance file access sites

Since it is expected that there will be a few different Object Adapters with interfaces that are appropriate for specific kinds of objects, time will tell if these are COTS to ECS or require specialized development.

3.3.3.7 Adapters to Legacy Services

Adapters, to make legacy services appear as objects will be required for some OO (and OMG)-based applications. The adapters will likely be implemented (via COTS) as part of the object services and/or common facilities, instead of a separate component. The major procedural

interfaces that will likely require encapsulation to provide OMG-based application extensions to well-known legacy services include:

- GSS-API (provides private and public key encryption, other security services)
- SNMP and CMIP/CMIS, possibly through XMP
- Directory, through DNS, CDS or XFN (CDS supports DNS and X.500 access)
- ftp
- smtp (possible X.400 integrated API encapsulation)
- telnet
- TCP and UDP sockets

The NASA V0 Network Office has numerous legacy services accessed by the aforementioned APIs. By building adapters to encapsulate these API's, the legacy services are readily accessed by advanced applications developed with the V1 infrastructure. The recognized V0 services and integration strategy include:

- listserv - (mail exploder) integrated with smtp adapter
- anon ftp - (file access) integrated with ftp adaptor
- X.500 - (directory) integrated with CDS or XFN adaptor
- email reflector - (mail forwarder) integrated with DNS/CDS/XFN and smtp adaptors
- Usenet, WAIS, WWW, Gopher - (information search and retrieval) integrated with a combination of the ftp and socket services
- V0 NOC services - (network mgmt) - integrated with XMP adapter

The proposed integration strategy has been successfully demonstrated by a number of vendors (HP, IBM, DEC) and consortia (OSF), and is an ideal candidate for evaluation prototyping.

Transitioning of V0 networks and routers is an transition planning issue not relevant to the architecture discussion. Potential IP address mapping issues relevant to ATM is discussed in the internetworking sub-architecture discussion of section 4. Additional management issues and strategies are discussed, in an architectural context, in section 5.3.

3.3.3.8 Traders and Federation

The concepts of service export, service import, and federation are explored in this section to better understand the information flow aspects of the communications sub-architecture.

3.3.3.8.1 Service Registration (export) Process

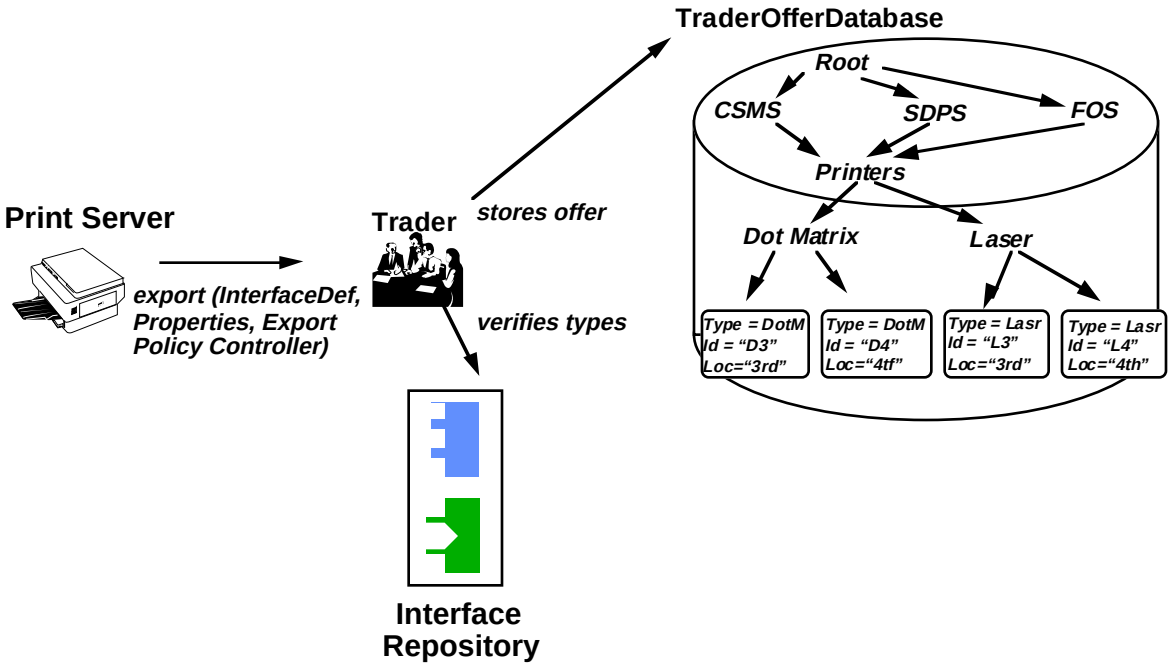


Figure 27. Service Export

Figure 27 highlights the service export process. Whenever a service exporter (server) wishes to advertise a service offer, it must register that offer with a trader. A service offer contains a description of the service it provides. A service offer is comprised of its interface definition (InterfaceDef), any properties and an optional export policy controller. Service properties can be static or dynamic. Static properties are properties which rarely change such as printer speed etc. Dynamic properties are properties that change frequently such as printer queue length. An export policy controller can optionally be designated by an exporter. Service offers are stored by the trader in a centralized or distributed database. The Interface Repository is utilized in this example.

3.3.3.8.2 Service Request (import) Process

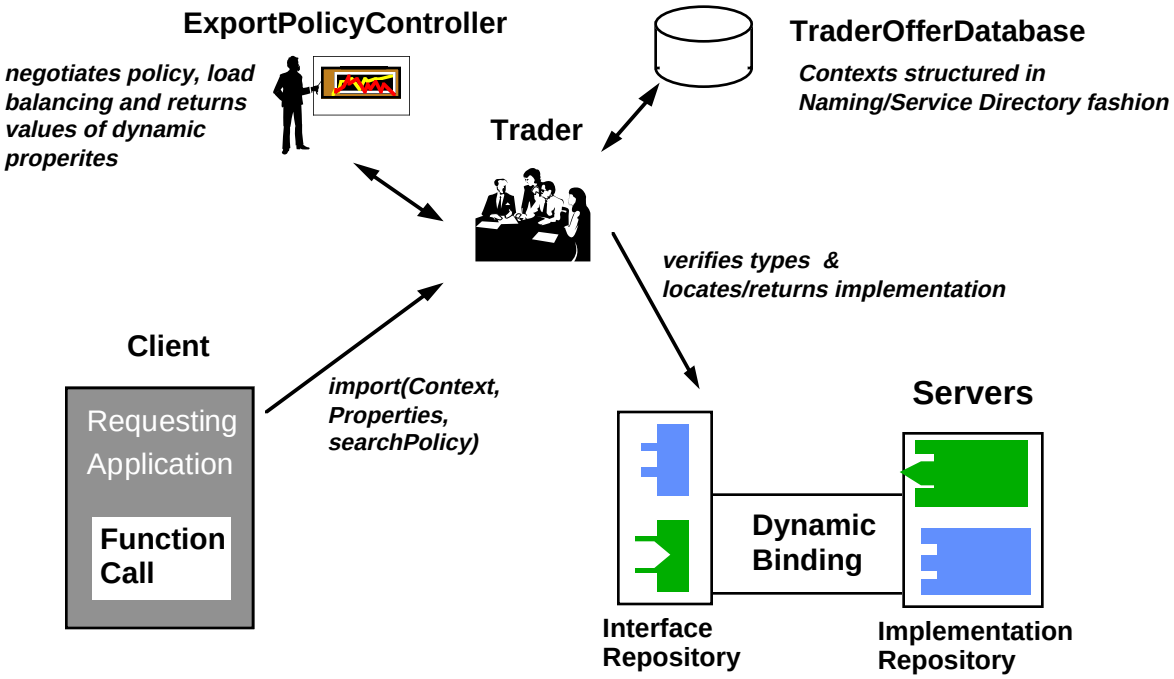


Figure 28. Service Import

Figure 28 highlights the service import process. Whenever importers of services (clients) require a service, a trader is contacted to accept requests. A service request is an expression of the requirements of a clients needs. A trader searches its offer database to find a matching service to fulfill the client's request. A trader can select the most appropriate offer or an export policy controller can utilize a services dynamic property value and select a service based on load balancing policy etc. A trader can return a list of offers or a single offer to the client.

3.3.3.8.3 Federation of Traders

The global offer space should be assumed to be extremely large. Scalability of the offer space is therefore critical. The offer space must be partitioned because no single trader can operate properly over the entire range of the offer space.

Some users may find that all of their service requirements can be fulfilled within the offer space of a single trader. Others users needs will demand expansion of the offer space. A tree structure describing the adjacency of traders must be available to properly broaden the search of the name space. Figure 29 depicts the chaining/referral of traders that can be implemented to expand the offer space. The policy implications of going beyond the immediate offer space must be understood and agreed to between trader administrators. A trader is just another service. By advertising itself to another trader, it opens itself up to interworking with the other trader and establishing a federation. The extent of what information they will share is largely policy - just as international trade partners establish contracts, so do the traders.

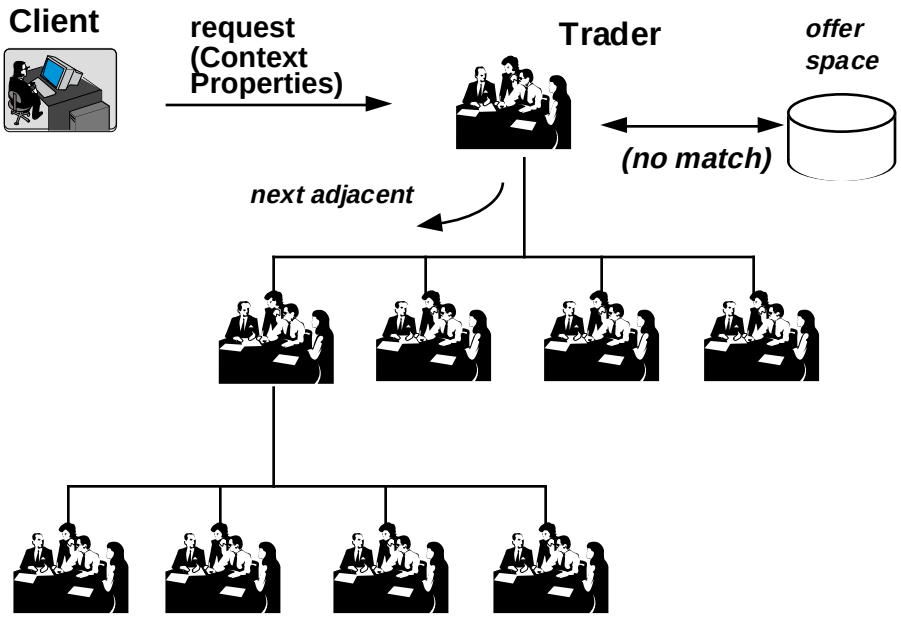


Figure 29. Federated Trading

3.4 Issues

3.4.1 Development Environment

Issues discussed in this section are intended to concentrate on the development of object-oriented management and service applications in the distributed, heterogeneous environment. This area will require ongoing SEPG analysis and oversight.

- Operating System - DCE technology was originally designed for UNIX platforms. Due to vendors acceptance of this technology, it has been ported to IBM OS/2 and MVS OSs by IBM, PC-DCE for Personal Computers (DOS), by Gradient, Window NT is porting DCE clients. Support for advanced operating system will need to be shown. Recently, the DEC/Microsoft bridge of CORBA to OLE using DCE's RPC as the wire protocol gave a good look of events ahead. CORBA, if the trader is successful, will allow heterogeneous implementations of middleware beneath the ORB, anyway.
- Debugging and other tools - The need for maximum computer resource utilization will require multi-threaded applications. Multithreaded debuggers are currently lacking. There are few vendors and universities working on developing multi-threaded debuggers(e.g. Texas A&M University, IBM-TORONTO etc.) that will need to be investigated.
- Quality of service considerations must still be taken into account while designing object-oriented distributed applications. Performance of the network, availability, and related QoS requirements must be understood.

- Security weaknesses - To achieve extensibility, many of the software interfaces in distributed systems are made available to clients. Any client that has access to the basic communication service can also access the interfaces to servers. Such an open architecture is attractive to system developers, but software security measures are needed to protect the services against intentional or accidental violation of access control and privacy constraints. DCE offers security, data encryption and capability-based access control. The issue is: is security being compromised on non-Unix platforms? HAIS is actively working with OSF in integrating the DCE kerberos services with MIT version 4 kerberos, so that computers outside of a DCE cell may be protected. Additionally, abstraction of DCE security into OMG security and the GSS-API offer an opportunity to make this security transparent.
- Exception and Event Libraries - If the distributed application does not contain any exception handlers and the application thread gets an error, the application thread is terminated and a system-dependent error message from the threads package is printed. These exception libraries are ported for all Unix platforms and OS/2, MVS and PC for system-dependent errors. Event libraries have a related, but more important requirement for the distributed handling of events. DCE provide event services to manage this type of issues and DCE threads have built in exception handling routines that CORBA products can use. Exception and event libraries must be mapped into the new development environment.
- Templates - Templates are an abstraction of collection of either an object or an interface or an action. The incremental modification relating templates must ensure that self-reference or recursion in the template of the parent class becomes self-reference or recursion in the template of the derived class. Object template subtype/supertype relationships do not necessarily coincide with behavioral compatibility. This can create an issue as to how templates can be easily used by the developer in a distributed environment, and if there are good tools/products available to use templates for the distributed object-oriented applications.

3.4.2 Interoperability Issues

Issues limiting or constraining interoperability include:

- Standard window environment for applications portability
- Standardization of C++
- Message catalogs mappings
- Error handling mechanisms

ORB user interoperability issues include definitions of interoperability and requirements for interoperability. Definitions of interoperability issues include discussion of anticipated usage scenarios and environments. Requirements and expectations for interoperability include identification of issues such as:

- Should all objects be interoperable or just some?

- Should a single ORB on most machine interoperate, or should many ORBs within a single machine interoperate?
- Is there a need to move objects between different ORBs?

ORB implementor interoperability issues include a general approach to interoperability, and potential mechanisms and protocols for interoperability. General approach issues discussions of whether a common interoperability mechanism directly connecting all ORBs is better or worse than a "universal ORB translator" that each ORB connects to. Specific mechanisms and protocol issues for interoperability include many items, such as:

- Inter-ORB registration protocols
- ORB-to-ORB handshaking protocols
- The role of the dynamic invocation interface of the ORB
- The role of the static invocation interface
- Requirements for extensions to existing ORB functionality and APIs
- Implications of the ORB to
  - name space
  - access control
  - persistent storage
  - data formats
  - static stubs
  - object references
  - message formats
  - authentication
- Connection of the ORB to alternative transport protocols

Many of these issues are currently being examined by OMG for the CORBA 2.0 specification release. Additionally, many of these issues can be readily resolved through the use of DCE infrastructural technology integrated with the ORB.

Other issues under analysis for resolution with CORBA 2.0 include support for other language bindings, requirements for additional object adapters beyond the basic object adaptor, operations and specifications for the interface and implementation repository, support for transactions and asynchronous communications, middleware support for multi-endpoint interaction and replication, concurrency, and OMG compliance certification and testability.

3.4.3 Quality of Service

QoS considerations are far reaching - well beyond the communications sub-architecture. Work is in progress in ISO to produce QoS architectures that would work for all forms of media and data. Within OMG and the use of a trader, part of the end-to-end service negotiation needs to include

negotiation of the QoS attributes for each service constituting a channel. QoS parameters are well known for the ISO-RM, and need to be revisited in light of multimedia technology, such as OMG CORBA and ATM. ISO/IEC JTC1/SC24 is investigating multimedia and hypermedia architectures in an RM-ODP framework. This may eventually be implemented using OMG CORBA and object services. QoS largely is a design issue and will be resolved past SDR. The QoS issues require articulation of QoS requirements as they pertain to relevant system performance and availability. Contributors working on QoS issues include ATM Forum, IMA, IETF, IEEE 802, NMF, and other groups.

3.4.3.1 Performance

The use of trader within the ECS architecture enables the specification of quality of service (QoS). QoS can be specified to improve user perceived performance. The implications of QoS are driven by system policy and affect information, engineering, and technology planning. For instance, if a policy decision is made for quick response time, then the information (data) model may require a special partitioning of its information (data), the engineering model may require replication of critical information, and the technology model may require broadband transmission of data, or bandwidth allocation capabilities.

Interfaces to the services defined in this document should allow the implementation of the service to be split into a low latency part and a latency tolerant part. The low latency part of the implementation could then be placed close to the user, while a latency tolerant part of the service could be placed anywhere. The concept of object caching, especially dynamic object caching, provides a good example of how specific records within directory, naming, and trading services, based on frequency of access, might dynamically be moved closer to the user to enhance performance.

A positive secondary effect of object caching, while improving performance, is to reduce WAN traffic due to the effective decrease of queries going outside the user's local environment. The economic benefits of QoS specification are therefore additionally emphasized. Additional discussion of QoS parameters described within the OSI-RM is provided in section 4.2 of this document.

3.4.3.2 Availability

Providing QoS criteria is largely an implementation issue. For instance, if a client is bound to a service, and a failure occurs, the interface to the service should provide for an alternate instance of the service to take over. This can be done by making the failure visible to the user, through some kind of a visual indication and the presentation of another interface to continue the service elsewhere. Alternatively, the capability might be hidden from the user, and a default kicks over to the new service for maximum availability, but the fundamental interface is unchanged.

3.4.4 Policy Implications

A trader is the center of a community established for the purpose of trading. The community consists of members (agents) that have roles such as: trader, exporter, importer, trading administrator, and trading policy maker. The trading activities of the community are service exports

and service imports. These activities are governed by a trader policy. Some important rules for the trader policy include:

- Security requirements for importers and exporters - rules to prevent unauthorized use, disclosure and modification of service offers, to detect security breaches, etc.
- Searching requirements - rules to govern the scope and strategy considered by a trader in a service import
- Accounting requirements - rules to determine charges for importing and exporting, credit certification, audit control, etc.
- Transfer requirements - rules for stating that importers can import service offers for their own use or pass them on to other entities; that exporters can export services they provide or services provided by other entities, etc.
- Trader Quality of Service requirements - rules for performance requirements, consistency requirements between what is given and what is wanted or between advertised and actual values, etc.

Some of the above requirements, e.g., security and accounting requirements, will be provided by other object services (such as access control lists within security, and systems management accounting services).

Each importer can also have its own importer policy that restricts the set of service offers considered by a trader in a service import. Each exporter can also have its own exporter policy that restricts the set of importers to which a trader may convey a service offer. On service import, the matching process performed by a trader is governed by the trader policy of the trader and the importer policy of the importer. On service export, the offer placement process performed by a trader is governed by the trader policy of the trader and the exporter policy of the exporter. The mutual matching of policies are called contracts, and can be drawn within ECS by Memoranda of Understanding (MoU's).

Every service within ECS is a potential importer or exporter. The policy implications of the service interactions need to be understood in light of the set of requirements listed above. As multiple traders are developed, a trading syndicate comprised of a community of trading communities may be established. This community, equivalent perhaps to a DAAC or the ECS, has a common administrator established for the purpose of making the service offer within each trading community available to other agents in the syndicate. When different administrators are involved (perhaps between DAACs or between ECS and other components of XnDIS), then a trading federation is established. In a trading federation, a community of trading syndicates might use their administrators to work with an external arbitrator established for the purpose of making the service offers within each individual trading syndicate available to other agents in the trading federation.

To ensure that security is maintained, while planning the access control list, it will be important for the ECS team to keep the level of access control restrictive enough. A special set of individuals or a special group should be given permission to create accounts and groups in the root directory of the security space. At the same time, enough access will be given to the individuals with administrative responsibility. Upon configuring cell, security, directory, time, file and other administrative

groups should be created to perform different administrative tasks for unique service classes. In addition to this groups, individual users need permission to control some information kept in the registry database, such as password, home directory, or login shell.

3.4.5 Incremental and EP Track Planning/Coordination

The phasing of the architectural components requires detailed coordination with the early CSMS developers, with an initial focus on the prototyping developers.

3.4.6 Multimedia Extensions

Multimedia technology is an emerging technology that ECS should anticipate as an architecture evolvability test. The IMA, through its MSS activities, should be publishing in October 1994 with MSS objects added as OMG object services. Applications exchanging multimedia data require support for streams of data whose characteristics differ from those currently supported by the CORBA specification. The media concerned can be classified as static (e.g., raster and vector images), sequential (e.g., animation), or isochronous (e.g., video or speech). The display of multiple media in a coordinated way requires synchronization between separate data streams (e.g., video and voice), and with user input. Integrating multimedia data into computer applications has several issues, including:

- How to control commonly-required manipulations of the data, such as splitting data from one source into several streams or mixing data from several sources
- How to synchronize related but separate streams, such as video and voice
- How to smoothly transition from multimedia data searching, to location, then to visualization

3.4.7 Market Forces and Competition

The object wars and vendor inter-firm rivalry are continuing factors in taking any position on any technology. There is good news - vendors are collaborating through consortia, and more recently, the consortia are talking to each other, and to the standards bodies. This movement is creating a better model of integrating market push with market pull. To combat the constant state of flux of the market, HAIS should actively pursue the following:

- High use of standards where applicable
- Consortia involvement
- Abstract, high level APIs that provide alternative middleware migration flexibility
- Continued use of subcontractor support for market and vendor analysis

HAIS has been successfully performing the tasks above to ensure up-to-date information of the market. The market dynamics should be welcomed as a sign of health in the industry to robust, viable implementations of advanced technologies for ECS. A short-term inconvenience has long-term benefits to ECS with better products.

4. Internetworking Sub-Architecture

4.1 Assessment of Needs/Drivers

The needs/drivers for the internetworking sub-architecture are an extension of the needs/drivers articulate in section 3.1. The internetworking architecture provides the local and wide area communications resource for separate logical processes, running on physical hosts, to share data with each other. Mechanism for the reliable end-to-end transfer of data from one communicating source to destination are the emphasis of this sub-architecture. Specific needs and drivers of the internetworking sub-architecture are to provide:

- a scalable network infrastructure
- an enduring network infrastructure
- a flexible network infrastructure
- an easy, low risk migration capability
- new capabilities, essentially "built-in"
- seamless LAN/WAN integration
- easy installation of operation
- high network availability
- the allocated use of shared network
- any-to-any connectivity
- geographic independence
- high performance, low delay
- bandwidth on demand
- seamless integration of technologies (e.g., shared vs switched mediums)

Many of these drivers are resolved with today's internetworking technology (largely through available TCP/IP-based infrastructure) and will not be discussed in detail in this paper. There are unique drivers not met with today's technologies, as well as architectural considerations of evolvability, that are not necessarily met with today's technologies. These areas include:

- architectural support for QoS end-to-end establishment
- Real-time protocols for file transfer, increased performance and multimedia
- scalability of network layer IP addresses and other fixed formatted protocols
- seamless technology integration of frame and relay technologies using ATM
- technology migration of routed flow to ATM switched fabric flows

QoS management is in the process of improvement through the introduction of OMG technologies and the use of a trader. Multiple instances of traders can exist for specific service types. Negotiation of QoS has occurred within protocol machines for years, and are only now being considered for logical distribution. This approach hold tremendous value for QoS negotiation for multimedia communications. Protocol improvements, including polymorphic formats, real-time allocations through QoS, and parallelism of state machine function through vertical partitioning are active areas of ongoing research around the country of which HAIS is reviewing and participating. ATM technology is an active area of research, consortia activity, and prototyping.

4.2 Logical Architecture

4.2.1 Introduction to OSI-RM

The ODP engineering model, discussed in section 3.2 and Appendix A, defines multiple transparencies that need to be 'hidden' from user/provider interactions. The internetworking sub-architecture is primarily responsible for providing platform-level communication resource transparency. The majority of the other transparencies are provided by services defined within the communications sub-architecture.

Internet-based communication systems today can best be modeled within the lower-level framework of the Open Systems Interconnection Reference Model (OSI-RM). The OSI-RM is a seven layer communications framework. The concept of layering within the OSI-RM includes the use of layers, communication entities, service access points, protocols, and connections, elements of layer operation, routing, and management. Of special relevance to the internetworking sub-architecture are the lower four layers of the OSI-RM - the transport, network, data link, and physical layers.

The principles of layering in the OSI-RM treats each open system as logically composed of an ordered set of multiple subsystems. In the case of the internetworking sub-architecture, there are four such subsystems - the transport, network, data link, and physical layers. Open systems intercommunicate between adjacent subsystems, on a layer by layer basis. The layers within a single open system communicate to the layers above it and below it.

Communicating entities exchange information by establishing an association in each particular layer using a protocol. There can be multiple classes of protocols defined for each layer. For instance, within the transport layer there may be a connection-based protocol, such as TCP , and a connectionless protocol, such as UDP.

Connection-oriented communication service at each layer involves connection establishment between two or more open systems, data transfer, and connection release. A clearly established lifetime is established with the following fundamental characteristics:

- two or more party agreements of the transmission of data among the peer layer entities using the provider of the lower-level service (for example, transport layer connection establishment through the use of the network layer service)
- negotiations between all parties of the parameters and options governing the transmission of the data

- address resolution and transmission overheads for connection establishment are avoided on the actual data transfer
- context for successive data transmissions are provided, making data sequencing and flow control possible

The characteristics of connection-mode transmission are attractive in applications calling for long-lived, stream-oriented interactions between entities, such as remote terminal, file transfer, and long-term remote job entry. Connectionless mode transmission is the transmission of a single unit of data from a source to a sink without connection establishment. The service is initiated by a single service access, and is inherently asynchronous. The characteristics of connectionless-mode transmission are:

- no dynamic agreement between the peer layers is required
- unit of data, destination address, QoS selection, options, etc. are transmitted together across the peer layer of the two open system entities.

Due to the asynchronous nature of connectionless-mode communications, it is possible that individual units of data are routed independently, and arrive in a non-structured order.

The elements of layer operations include protocol selection and identification, protocol version selection and identification, negotiation mechanisms, and communication functions. For each layer, protocol identification is a process to determine the type of protocol being used. Multiple protocols may be used within a layer. To enable communication, however, there must be an agreement of the particular protocol. The protocols are identified either by the protocol itself, using an embedded protocol identifier, or by an address map that identifies the protocol at a particular entity. The protocol version identifier identify the level of a particular protocol in use (for instance Kerberos V4 vs. Kerberos V5). The identification of the version presupposes that the protocol has been identified. Provisioning for alternative protocols or protocol version provides the flexibility, within a layer, to substitute in alternative protocols.

The negotiation of protocol version can only occur in connection-mode communication. The calling entity, at a particular layer, sends information of all supported protocol versions to a peer entity. The called entity checks for a compatible version,and uses the latest version available to both peer entities. If there is no common version, the connection establishment request is rejected. Connectionless-mode protocols provide no negotiation mechanism. Identification of the protocol is either known a-priori, or explicitly conveyed within the protocol data unit 's (PDU's) transmitted.

The functional modes of communications for connection and connectionless communications are identified in Table 3 below. Note that not all modes are required for a protocol implementation. The OSI-RM provides definitions of each of the functions for the interested reader.

Table 3. Communication Mode Functions

Function	Conn	C'less
Conn Estab. and Release	X	
Suspend	X	
Resume	X	
Muxing & Splitting	X	X
Normal Data Transfer	X	X
Data Transfer during Estab.	X	
Flow Control	X	X
Expedited Data Transfer	X	
Segmenting	X	X
Blocking	X	
Concatenation	X	X
Sequencing	X	X
Acknowledgement	X	X
Error Detect & Notification	X	X
Reset	X	
Routing	X	X
Quality of Service	X	X

The function names are largely self-explanatory with the possible exception of expedited transfer of data, routing, and quality of service functions of communications. An expedited transfer of data is a data unit transmitted to a peer entity with priority transmission and/pr processing over normal data units. An expedited data transfer is typically used for signalling and interrupt purposes, and is independent of the states and operations of the normal data flow. The flow is assumed to be used to transfer small amount of data infrequently, and is not to be considered a part of routine transfer of data. It is available as a means to handle exceptional circumstances. Routing within a layer enables communications to be relayed by a chain of peer entities. The routing function is transparent to other layers in an open system. The layer entity which participates in a routing function usually has a routing table, such a routing device for the IP layer.

Quality of Service (QoS) is a collective name given to a set of parameters. There are two sets of QoS parameters. These parameters are negotiated at the time the connection is established, either by the peer layer entities, or by an intermediary. The first set applies to both connection and connectionless service and includes both single- and multiple-transmission parameters. The second set of parameters are for connection-mode communications only. QoS parameters are defined for each layer of the OSI-RM. A sample of the parameters is provided below:

- Connection/Connectionless Parameters
  - Single Transmission Parameters
    - expected transmission delay
    - probability of corruption
    - probability of loss or duplication

        - probability of wrong delivery
        - cost (based on number of hops or shortest path)
        - protection from unauthorized access
        - priority
    - Multiple Transmission Parameters
        - expected throughput
        - probability of out of sequence delivery
- Connection-Mode Parameters
    - connection establishment delay
    - connection establishment failure probability
    - connection release delay
    - connection release failure probability
    - connection resiliency

Figure 30 illustrates the OSI-RM. The internetworking sub-architecture only extends through to layer 4 - the transport layer. The communications sub-architecture, discussed in Section 3, occupies layers 5-7 of the OSI-RM, but does not adhere completely to a layered architecture. As such, services defined within the communications sub-architecture actually span multiple OSI-RM defined layers. The minimalization of layers beyond the transport layer is desirable to improve performance and minimize overhead. Vertical partitioning, instead of horizontal layering, of communication services above the transport layer enables evolvability and separation of components for QoS optimization.

The internetworking architecture is comprised of end systems and intermediate systems. End systems include end-to-end (source-to-sink) data transmission over the transport layer, and layers one through four, at a minimum. They are open systems which, for a particular instance of communications, are the ultimate source or destination of data. Intermediate systems , also called relay systems, are open systems that make use of OSI functions up to a certain layer, providing a relay function at the highest layer supported. Figure 30 depicts a network layer relay, generally called a router. End systems may communicate to other end systems through the services of intermediate systems. A general description of the transport, network, data link, and physical layers of the OSI-RM and associated services follows.

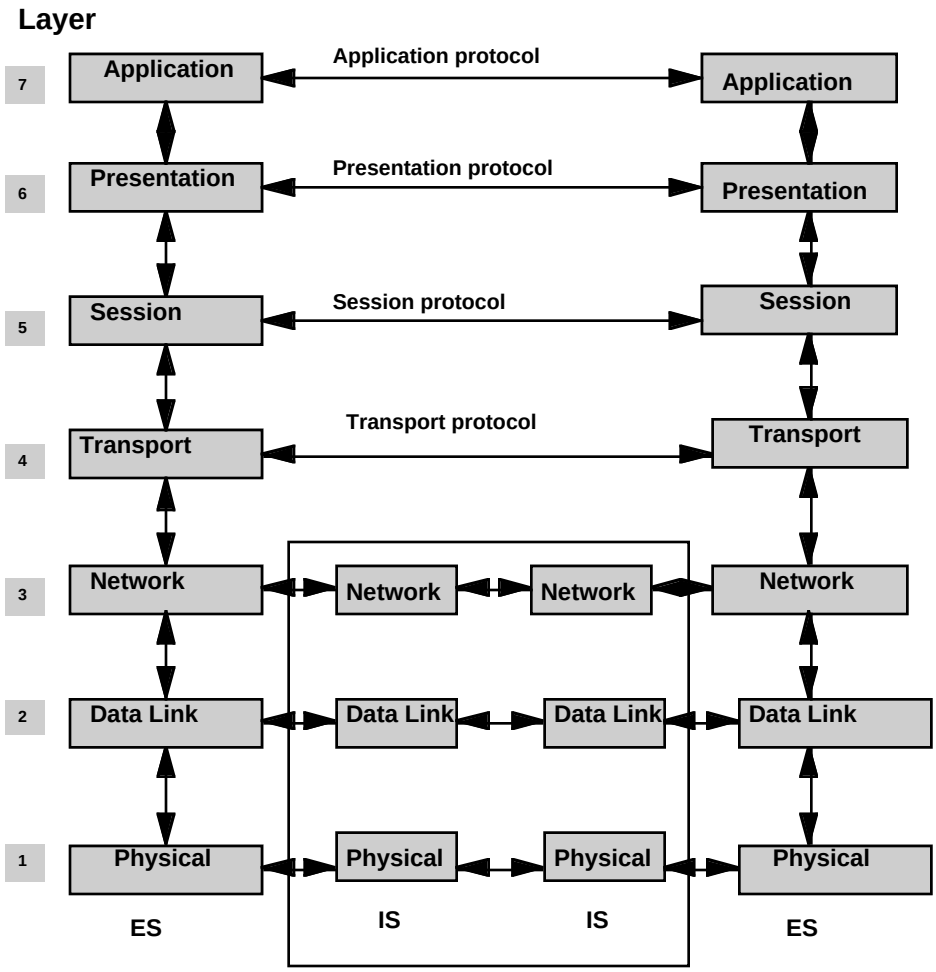


Figure 30. Communication Involving Relay Open Systems

4.2.2 Transport Layer

The transport layer provides transparent transfer of data between higher level entities and relieves them from any concern with how reliable and cost effective transfer of data is achieved. Cost/performance optimization is achieved within constraints imposed by the overall demands of all concurrent higher level sessions, and the overall quality and capacity of the network-service available to the Transport layer. All protocols defined in the transport layer have end-to-end significance. This means that the transport layer is end system oriented and that transport protocols only operate between end systems. The transport layer is not concerned with routing and relaying aspects - this is a lower-level issue. To provide a requested service quality, the transport functions invoked in the transport layer depend on the quality of the network layer service.

Primary services provided by the transport layer for connection-mode communication include transport connection establishment, transport connection release, data transfer, expedited data transfer, and suspend. Connectionless-mode communication provides all of the above except for data transfer omits segmentation and PDU reassembly.

4.2.3 Network Layer

The network layer provides the functional and procedural means to exchange network data units between transport entities over network connections, both for connection-mode and connectionless-mode communications. It relieves the transport layer from concern of all routing and relay operations associated with network connection. The basic function of the network layer is to provide the transparent transfer of data between transport entities.

This layer contains all functions to provide to the transport layer a firm network/transport layer boundary, independent of underlying communications media in all things other than QoS. In this way, the network layer masks the differences in the characteristics of different transmission and subnetwork technologies into a consistent service.

The QoS is negotiated between the transport entities and the network service at the time of establishment of a network connection. While the QoS may vary from one network connection to another it will be agreed for a given network connection and be the same at both network connection points. This QoS will be maintained for the duration of the connection. Specific QoS parameters include residual error rate, service availability, reliability, throughput, transit delay (including variations), and delay for network connection establishment.

4.2.4 Data Link Layer

The data link layer provides functional and procedural means for connectionless-mode communication among network entities, and for connection-mode communications for the establishment, maintenance, and release data link connections among network entities, and for the transfer of data link data units. A data link connection is built upon one or several physical connections. Detection and possible correction of errors occuring in the Physical layer is taken care of by the data link layer. Additionally, the data link layer enables the network layer to control the interconnection of data-circuits within the physical layer.

QoS parameters may be optionally selected. The data link layer establishes and maintains a selected QoS for the duration of the data link connection. The QoS parameters include mean time between detected but unrecoverable errors, residual error rate, service availability, transit delay, and throughput.

4.2.5 Physical Layer

The physical layer provides the mechanical, electrical, functional, and procedural means to activate, maintain, and deactivate physical connections for bit transmission between data link entities. A physical connection may involve intermediate systems, relaying bit transmissions within the physical layer. Physical entities are interconnected by a physical medium in the physical layer. The majority of the LAN/WAN channel bit transmission rates are depicted in Figure 31. The average

available bandwidth available to a physical node at any given time is based on an equal access channel with no priority of users. The diagonal lines in the chart represent shared media, where average available bandwidth drops as the number of communicating nodes increases. The vertical lines represent full bandwidth availability to the desktop introduced by non-blocking switched media. Bus and point-to-point technologies are not shown.

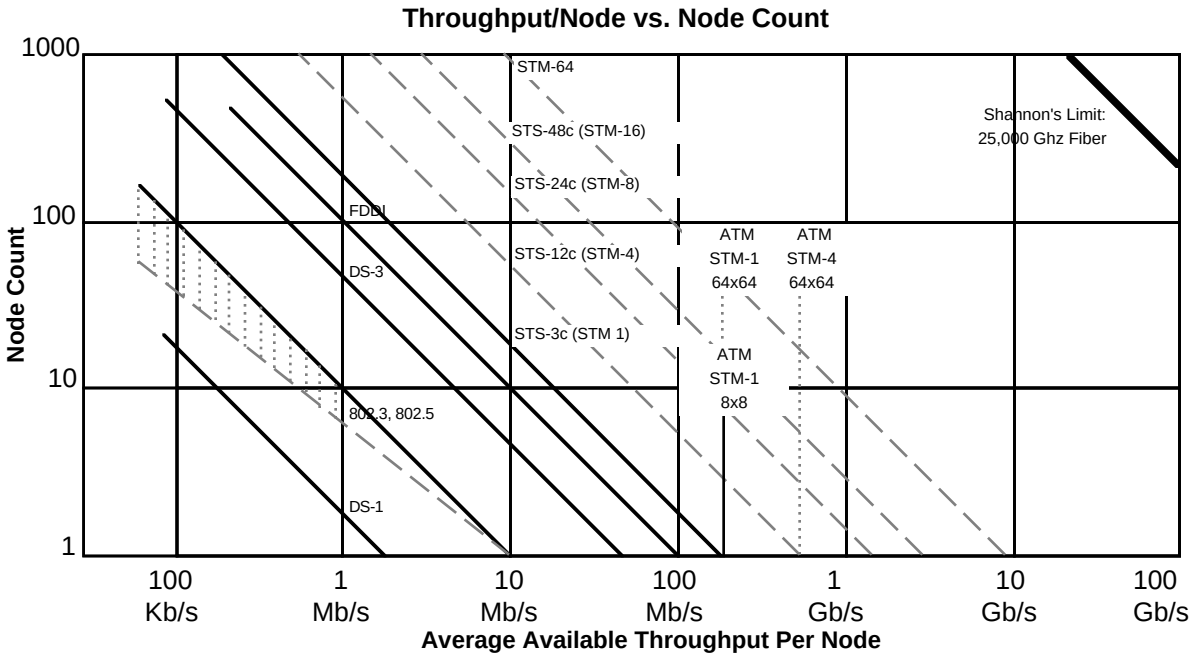


Figure 31. Topology Considerations for Physical Layer

4.3 Implementation Architecture

4.3.1 Internetworking Services

The major internetworking service classes are transport layer service, network layer service, data link service, and physical layer service. These correspond directly to the logical architecture discussed in section 4.2, and provide the same essential services as previously discussed. This section examines alternative protocols for each service class. Further analysis of the protocols is not explored.

4.3.2 Transport Services

Entities within the communications sub-architecture specify the class of transport service to be provided at connection establishment. The transport service classes can be characterized by combinations of selected values of parameters such as throughput, transit delay, and connection set-up delay. Guaranteed values of parameters may additionally be requested, such as residual error rate and service availability. All of these parameters are QoS parameters that are negotiated prior to connection establishment, either by the transport layer service itself, or by an intermediary such as a transport-type trader. In selecting the range of transport services required, it is imperative

that all transport service requirements for the various types of anticipated traffic flows generated are covered.

Interaction of the transport service to other service layers is well defined in the OSI-RM. Implementations of the transport layer service include (non-exhaustive list):

- IETF Transmission Control Protocol (TCP)
- ISO TP4 (ConnectionLess Transport Service)
- IETF User Datagram Protocol (UDP)
- ISO TP0, 1,2, & 3 (Connection-Oriented Transport Service)
- TCP Extensions for High Performance
- XTP experimental

ISO has defined five classes of transport protocol: Type A network (Class 0 and 2 - simple and multiplexing class), Type B network (Class 1 and 3 - basic error recovery and multiplexing class), and Type C network (Class 4 - error detection and recovery class). The transport service specification is the same for all classes. In Europe Type A network with Class 0(often called TP0) is very popular, because of it's simplicity and good performance.

Transmission Control Protocol/Internet Protocol (TCP/IP), which is connection oriented, is designed to provide reliable communication between pairs of processes across a variety of reliable and unreliable networks and internets.This stream oriented protocol provides two useful facilities: Data stream push and Urgent data signaling. TCP connection establishment is a three-way handshake. To initiate a connection, an entity sends an RFC (request for connection) X, where X is the initial sequence number. The receiver responds with RFC Y, ACK X by setting both the SYN and ACK flags. Finally the initiator responds with ACK Y. Data transfer in TCP is viewed logically in terms of stream of octets. TCP normally exercises its own discretion as to when to construct a TPDU for transmission and when to release received data to the user. The normal means of terminating a TCP connection is a graceful close. An abrupt termination occurs if the user issues an ABORT primitive. In this case, the entity abandons all attempts to send or receive data and discards data in its transmission and reception buffers.

The User Datagram Protocol/Internet Protocol(UDP/IP) provides a transport level datagram service. UDP is basically an unreliable service; delivery and duplication protection are not guaranteed. To establish the connection, UDP assembles a data unit and hands it to IP for transmission. Incoming data units are checked using the checksum, but there is no provision for error reporting.

The OSI Class 4 transport protocol (often called TP4 or Type C network) and TCP have numerous similarities but also some differences. Both protocols are designed for providing a reliable, connection-oriented, end-to-end transport service on top of an unreliable network that can lose ,garble, store, and duplicate packets. Both must deal with worst case problems such as a subnet that can store a valid sequence of packets. TP4 and TCP both use three-way handshakes and share the general concepts of establishing,using and releasing connections. As listed in Table 4, both protocols have notable differences.

Table 4. TCP and TP4 Comparision

Feature	TCP	TP4
Number of TPDU types	1	9
Connection Collision	1	2
Addressing Format	32 bits	not defined
Quality of Service	Specific options	open ended
User Data in CR	Not permitted	Permitted
Stream	Bytes	Messages
Important Data	Urgent	Expedited
Piggybacking	Yes	No
Explicit Flow Control	Always	Sometimes
Subsequence Numbers	Not permitted	Permitted
Release	Graceful	Abrupt

Further analysis, tradeoff, and selection of specific implementation(s) is not discussed in this paper. The TCP transport is the protocol of preference for ECS, however, based on its dominance in industry usage and wide-community support.

4.3.3 Network Services

The network service classes can be characterized by combinations of selected values of parameters such as throughput, transit delay (including variations), and delay for network connection establishment. Guaranteed values of parameters may additionally be requested, such as residual error rate and service availability. All of these parameters are QoS parameters that are negotiated prior to connection establishment, directly by the network layer service through one or more protocol implementations.

Interaction of the network service to other service layers is well defined in the OSI-RM. Implementations of the network layer service are generally suballocated to optimize specific functions within the network layer service, such as routing, address resolution, and network data unit transfer. Implementations of the network layer (sub)services include (non-exhaustive list):

- Internet Protocol
- Internet Control Message Protocol
- Open Shortest Path First Routing Protocol
- Address Resolution Protocols (ARP/RARP)
- Routing Information Protocol
- ISO Connection-Oriented Network Protocol
- ISO Connectionless Network Protocol
- End System Intermediate System
- Intermediate System Intermediate System

- TUBA, IPng, SIP, PIP, IPAE, GUP, other experimentals
- S2K Real-Time IP

Further analysis, tradeoff, and selection of specific implementation(s) is not discussed in this paper. The internet protocol (IP) is the protocol of preference for ECS, however, based on its dominance in industry usage and wide-community support.

4.3.4 Data Link and Physical Layer Services

There are many implementations of data link and physical layer services. The more popular include 8802.2 LLC with 8802.3, 8802.4, or 8802.5 MAC for the data link layer, and 10 BaseT, and FDDI PLP for the local LAN physical layer vice SMDS, Frame Relay, Synchronous Digital Hierarchy for the WAN physical layers. Recently, an integration of LAN and WAN technology has been promised with ATM technology. ATM may prove to be the preferred internetworking solution (see Figure 35) in the next three to five years, but many problems remain unresolved. These issues are addressed in section 4.4. Architectural aspects of the data link and physical layer services revolve more around policy requirements, modeling analysis, and QoS requirements then high-level architecture and will be the subject of ongoing work through PDR.

4.3.5 Network Security Architecture

Security architecture is a complex issue, dependent on many interrelated issues, included QoS and policy requirements, and design implementation details. A security design is due to the customer at PDR + 2 months. Specific security mechanisms in consideration span all three CSMS sub-architectures and is shown in Figure 32.

Access control, authentication, data integrity, and encryption are all related to the communications and systems management sub-architecture services. The access control mechanism is used to discourage unauthorized access to a resource or to limit the use of a resource in an unauthorized manner. Authentication exchange is a mechanism wherein the identity of a party must be verified before access is granted to a resource. Data integrity is a mechanism that is used to ensure that data has not been destroyed or altered in an unauthorized manner. Encipherment (encryption) uses cryptographic techniques to encrypt data, either automatically via software, or via smart cards carried by 'secure' personnel. The policy implications of security, and systems management security overall are discussed further in section 5. The combination of access control, authentication/authorization, data integrity, and encryption together affects cell engineering practices and the logical distribution of services.

Digital signature, traffic padding, notarization, transport and network level encryption, and routing control are all internetworking related security mechanisms. Digital signature is used to ensure that the recipient of data is the proper recipient and also to ensure that the data has not been changed. This is done by applying cryptographic information to a protocol data unit. Traffic padding is a mechanism wherein spurious bits, octets, or other blocks of data are appended to protocol data units.The notarization service ensures that a third party is used to guarantee the accuracy of information, not only its content but its proper origin, timing, and delivery as well. Routing control contains rules that allow relay mechanisms to avoid specific networks or data communications

links for purposes of security. The use of dual-homing and router firewalls have both security and policy implications.

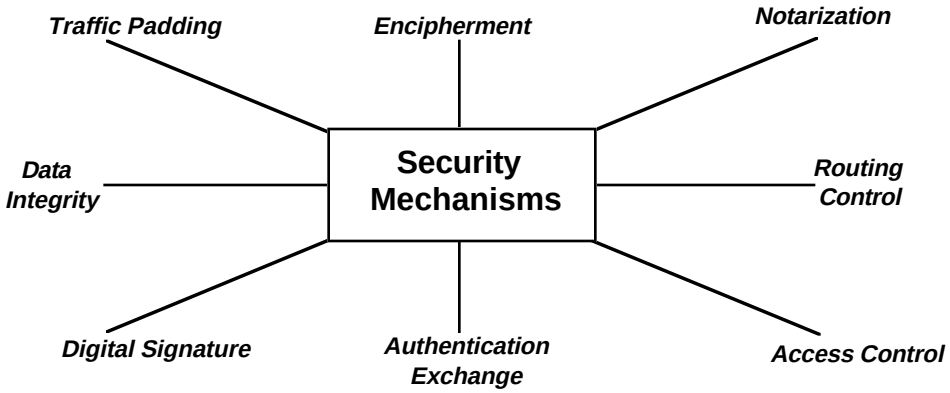


Figure 32. Security Mechanisms

4.3.6 Network Topology Architecture

The detailed aspects of LAN/WAN sizing and configuration is a design implementation process not discussed in this paper. The architectural impacts of LAN and WAN design have a high degree of coupling with policy requirements and the SDPS Archive and Processing work in progress that will in a large way drive the resultant recommendations for LAN/WAN design.

Modeling will be performed as documented in the modeling plans to size input/output flows for each site. This work will be integrated with site specific processing and archive placement to arrive at a recommended implementation.

Figure 31 (previously illustrated) illustrates alternative shared and switched media topology alternatives from a capacity perspective. The three-tier media decomposition in the HAIS proposal to NASA is still valid for topological design. There is a natural classification of media into buses, switched media, and shared media. Each class of media will be analyzed in light of its ability to meet the loading and QoS requirements required at each site.

4.4 Issues

4.4.1 Performance Issues

4.4.1.1 Delay

The large file sizes of ECS datasets and the potential for significant distances between users and providers creates a time/bandwidth phenomenon that needs to be carefully considered for WAN pipes sizing. For very large file sizes over relatively short distances, the bandwidth of a pipe may appear to be the constraining factor in the time it takes for a file to get completely transferred from one place to another. The electrical delay of the pipe is negligible compared to the actual time it

takes for a file to be transferred due to bandwidth constraints because, in effect, the file is absorbing the total capacity of the communication channel. By increasing the bandwidth, the transfer time is improved, but only to the point where the electrical delay of the communication path is less than the quotient of the file size against the allocated bandwidth. At this point, the file can be totally absorbed by the pipe, and capacity for additional data would still be available. Now the response time is limited by the electrical delay of the pipe. In other words, dependent on the file size, the limiting factor in file transfer time can alternate from being constrained by bandwidth to being constrained by the absolute electrical path delay. Initial response time will always equal the electrical path delay (this is the time it takes to get one bit of information through) plus the transfer setup time, but time for total file transfer is not necessarily improved by adding more bandwidth.

4.4.1.2 Object Caching

The concept of object caching has been around for a relatively long time. Advanced research is investigating ways to dynamically allocate object stores based on user frequency of access to the data within the object stores. Key stores for the ECS include directory, security, file, and trader stores. By intelligently placing information, user-perceived performance is improved, and WAN object store access dramatically drops. This strategy should be exploited as much as possible to reduce internet traffic *and* improve performance. HAIS is currently negotiating a contract with the University of Southern California (USC) to further investigate this issue.

4.4.1.3 Protocol Enhancements

Many universities and research initiatives (CNRI) are investigating methods to improve the performance of transport and upper layer protocol implementations. Some of the methods include VLSI implementation, alternative code implementations, and the vertical partitioning of state machine functions into latency tolerant and latency intolerant components. HAIS has had discussions with CNRI on this subject and is planning an RFP on this topic this area for further analysis. Related topics of protocols are discussed in 4.4.4.

4.4.1.4 Quality of Service Mappings

Total QoS negotiations at all levels of a communications stack, while specified in the OSI-RM, is only partially implemented in practice. With the future inclusion of OMG CORBA and Object Services, and ATM technology, an opportunity exists to integrate end-to-end QoS negotiation for the entire user to provider channel. The use of OMG trader in this area is vital to act as a liason in coordinating all subsystem entities.

Integration of the trader with ATM QoS parameters is an ongoing area of work both within ATM Forum and the vendor companies.

4.4.2 Multimedia Issues

In the future, applications which exchange multimedia data will require support for streams of data whose characteristics differ from those currently supported by the CORBA specification. The media concerned may typically be classified as static (e.g., images), sequential (e.g., animation), or isochronous (e.g., video or speech). The display of multiple media in a coordinated way

requires synchronization between separate data streams (e.g., video and voice), and with user input. Integrating multimedia data into computer applications raises a number of issues, such as:

- How to control commonly-required manipulations of the data, such as splitting data from one source into several streams or mixing data from several sources
- How to synchronize related but separate streams, such as video and voice.

The interactive multimedia association (IMA) is involved with analysis of this problem and is developing CORBA-based common facilities for multimedia applications. Additional, ISO is developing a framework architecture with QoS negotiation for multimedia applications.

4.4.3 ATM Issues

ATM uses the mechanism of multiplexing variable-sized datagrams onto small fixed-size cells. Cells offer performance and implementation advantages to networks that service many types of traffic, but they incur bandwidth inefficiencies due to protocol headers and datagram fragmentation. The strategic issues associated with ATM, as with any new technology, are individual and tied to economics vs. productivity as shown below:

- Does traffic load really require migration from existing solutions ?
- What kind of ATM migration plans do vendors offer ? (Is this a killer superhighway or a technology looking for the market?)
- If ATM really is the next generation of technology, should one wait for the price drop?
- How many existing network devices (e.g. router, gateway etc.) will be compatible with ATM?
- From cost point of view, is frame relay better or ATM?

Technical issues of ATM still unresolved by the industry are highlighted below.

4.4.3.1 Traffic Management

Studies have shown that cell-based networks, using standard protocols are inefficient in carrying wide-area data traffic. ATM-based networks using SMDS and IEEE 802.6 protocols lose more than 40% of their bandwidth to overhead at the network level and below. Control actions are needed to improve the traffic performance during overloads and ATM network failures. High-performance ATM based networks do, at times, experience failures, congestion, and periods of degraded performance. During this period of excessive congestion, a switch could begin selectively discarding ATM cells, with the intent of maximizing delivery of high priority traffic by setting the ATM cells with the priority bit to zero for as long as possible. During congestion periods, ATM switches also could transmit ATM layer congestion notifications in both the forward and backward directions of virtual channel connection (VCC) and virtual path connection (VPC) transmission. ATM cells contain operations information such as a performance data, for the underlying ATM connections. These cells are referred to as operations and maintenance (OAM) cells. Use of these cells for reporting congestion (as opposed to simple cell tagging) allows for additional information such as level and cause of congestion, to be supplied along with the

congestion indication itself. Operations and management play a key role in the viability and success of the emerging ATM networks.

4.4.3.2 Address Mapping: LAN/WAN Adaptation and Interfaces

Address mapping of the IP address to ATM addresses are not well defined. This area will need to be addressed in future work by the ATM Forum and standards bodies. The issue can be extended to all ATM Adaptation Layers (AAL) for LAN and WAN integration and interoperability. Open bridging is a technology solution to this problem, but it is not been proven operationally.

4.4.3.3 Signaling and Call Control

ATM represents a new generation of switched architectures, adding high-speed, low-latency, end-to-end management and supporting multimedia applications. Because of uniform packet size, traffic is switched very fast through a connection-oriented network. By contrast, switched 56, ISDN, SMDS, HSSI (High-Speed Serial Interface), HIPPI(High-Performance Parallel Interface) require more addressing overhead and use slower technology; frame relay and X.25 have longer latency and carry only data.

A comprehensive signaling and connection management system will be critical to operating an ATM network. At higher transfer rates, by the time a problem can be detected and an adjustment made, the damage may already be done. Connections must be capable of being brought up, or taken down, as needed to reflect dynamic traffic patterns and application QoS requirements. ATM Forum is working on the Q.93B signalling standard to help resolve this problem, but it has not been tested with WAN delays introduced.

4.4.3.4 QoS Integration with Trader

This issue was referenced in 4.4.1.4. QoS mechanisms to manage overall traffic flow and isolate individual flows with specific QoS requirements is essential to the success of ATM. The QoS service must scale over a large range of data rates and network facilities.

4.4.4 Fixed Format Protocols

All fixed format protocols, TCP, IP, etc. used a fixed length PDU that has control bytes allocated for the different protocol functions. Certain functions do not scale well in a larger scale computing environment. Packet sizes, window sizes, source and destination port assignments, sequence count, and address are all functions that ultimately do not scale with fixed format protocols. The waning supply of Class C address space allocations for IP is a testament to this issue. ECS performance over long distances could become constrained by TCP sequence and window sizes. The 64 Kbyte packet size is very small in consideration of the file size estimates associated with ECS. Two byte source and destination port assignments are limiting the number of accessible applications on any given host. Either new protocols with longer fields will eventually be required, or protocols with more flexible formats are in order for the next generation of protocols.

5. System Management Sub-Architecture

Note: a separate white paper discussion of the systems management services exist that discusses the policy aspects of systems management relevant to service distribution of fault, configuration, accounting, performance, security, and other primary systems management services. This paper, and specifically this section on systems management, focuses on the policy-neutral architectural aspects of systems management.

5.1 Assessment of Needs/Drivers

At the ECS System Requirements Review (SRR), it was made clear that a perception existed in the community that the SMC was a centralized solution to systems management that represented a single point of failure to the ECS. Furthermore, great concern was expressed over the presentation of operational scenarios that portrayed the SMC as a deterrent to user access and a burden of authority. Regardless of any perceptions, the systems management sub-architecture has the following needs or drivers:

- no single point of failure
- allow DAACs to manage their own resources
- provide system-wide monitoring and coordination
- provide a policy neutral architecture
- within an implementation, allow distribution of authority based on policy
- do not interfere in-line with operational DAAC functions

Avoiding any single point of failure is paramount to good system engineering practice. All system functions must be implemented in a manner that provides high availability and failure protection. The concept of a central monitoring and coordination function (versus central management and control) allows GSFC to monitor activities between all DAACs and within DAACs to identify problem areas and coordinate solutions as required. Performance analysis of the total system would become a predominant SMC function. Flows outside of ECS would be monitored at the SMC.

Local management at the DAACs of their unique resources, with visibility when required across the system is synonymous with the concept of federation in advanced enterprise management solutions. The federated approach to systems management allows the DAACs to work problems between themselves without having to get GSFC involved unless appropriate. This represents an expedient approach to problem resolution, by allowing the involved parties to directly resolve a problem.

A policy neutral architecture defers policy decisions out to the implementation (design) stage, and allows flexibility for system reconfiguration to reflect policy changes over time. Distribution of authority provides flexibility for monitoring and coordination backup, both at the DAACs and at

GSFC. Finally, the concept of systems management being unobtrusive to in-line operations acknowledges that systems management is a *service* to users.

All of these needs/drivers can largely be met with COTS and near-COTS products that will be discussed in this section of the CSMS architecture working paper. The architecture is based on emerging enterprise management solution that are based on CORBA 1.1 technology with the backing of major consortia and industry vendors.

5.2 Logical Architecture

Figure 33 illustrates the systems management logical architecture. Systems Management can logically be viewed as being comprised of managers and managed objects arranged and interconnected through a protocol and a management information model. The manager has management applications, communication services, and an information model. The managed object shares communications services with the manager, and adheres to selected parts of the information model.

The management applications are popularly recognized as fault, performance, accounting, configuration, and security management. These application classes actually belong to the management functional areas defined in ISO IS 7498-4. These high-level application classes (called functional areas in ISO) are in a state of evolution and due to be finalized sometime in 1994. Additional high-level systems management applications are recognized, both throughout the industry and on the ECS project. The ISO functional areas actually rely on numerous underlying system management functions (SMF's). These underlying management services are similar to, and have an overlap with, many of the object services and common facilities identified for the CORBA architecture of section 3.3. Section 5.3 of this paper will discuss this topic in more detail.

The communications services include the mechanisms to transfer systems management related traffic between managed objects and the manager, or between managers. The communications between a managed object and a manager involves an agent, typically co-resident with the managed object, transferring information through a communications channel to the manager. The agent collection mechanisms, notification mechanisms, and manager operations mechanisms are all part of the communications service between the manager and the managed object. The collection mechanism is involved in retrieving and maintaining a local instance of data about the managed object. The notification mechanism is used to report events about managed objects. The operations mechanism defines the operations to create, retrieve, modify, delete or perform other operations on an agent. The interaction between the manager and a managed object additionally involves a protocol. The protocol provides the actual transfer service for the management application and agent, with appropriate interface to the underlying communications infrastructures, as detailed in section 3.3 and 4.3 of this paper. Communication service interaction between managers typically involves just the notification mechanisms and the protocol.

The management information model defines management information flowing between managed objects and the manager, and is used as a template to create agents to the managed objects. The attributes of a managed object that can be manipulated by the manager are defined in the information model, as well as the operations of the attributes and the operations that may apply to the managed object itself. Managed objects, events, attributes, operations, and rules for the

creation of templates to be used in the creation of managed object agents are all defined by the management information model.

The manager has many applications with which to monitor and configure systems resources (managed objects) as required. The managed objects, through the services of an agent, react to manager directives or requests to change the behavior of the managed object. The system works because all managed objects, the protocols to talk to agents, and the specific information to be shared are defined a-priori. The arrangement of the managers and managed objects usually take on one of two predominant logically distributed architectures: either a hierarchical distribution or a federated distribution.

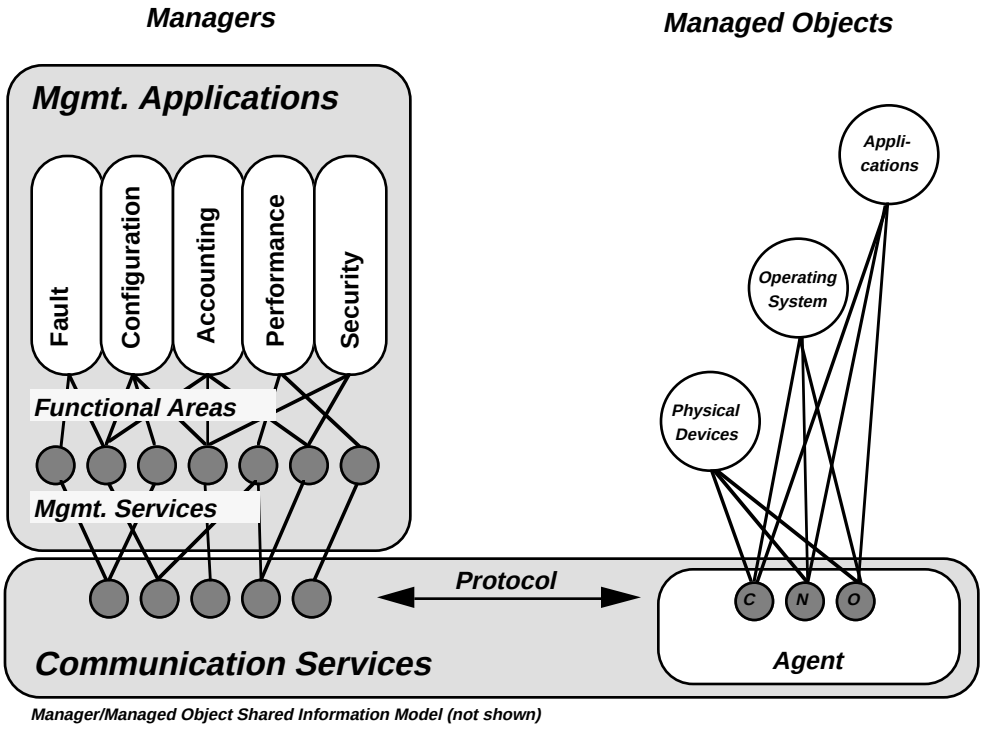


Figure 33. Systems Management Logical Architecture

5.2.1 Hierarchical Architecture

Figure 34 illustrates a hierarchical network of managers. In a hierarchical architecture, managed objects typically interact with a single manager. The exchange of traffic is between the managed objects and the manager. The management structure revolves around a management application with remote “agents” all reporting to it. The structure is fairly rigid, and does not scale well to larger management domains without adapting levels of hierarchy. The 'manager of managers' resolves problems across the multiple management domains of the lower-level managers. Lower level managers do not talk to each other without invoking the services of the higher-level manager.

This approach to systems management characterizes many systems in place today. The available technology has largely forced this approach to systems management, and does not satisfactorily address the needs/drivers discussed in section 5.1. There are several severe widely acknowledged deficiencies with a hierarchical architecture approach. The architecture places obvious burdens on the highest level manager when lower-level managers cannot talk to each other for problem resolution - causing delays and potential misrepresentation in problem resolution. A global namespace at the top of the hierarchy must be maintained for overall system management to be effective. The hierarchical nature of the architecture implies policy by the designation of a central authority to oversee and control all system functions. The highest level manager can be viewed as a single point of failure to the system. Although redundant hardware and distribution/replication of critical files stores are provided with today's system's, spatially the highest level manager is not protected.

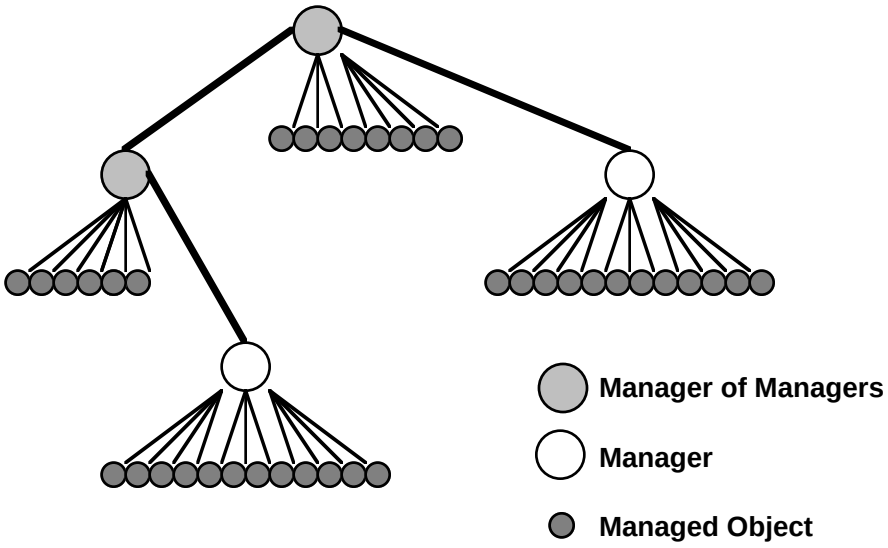


Figure 34. Systems Management Hierarchical Architecture

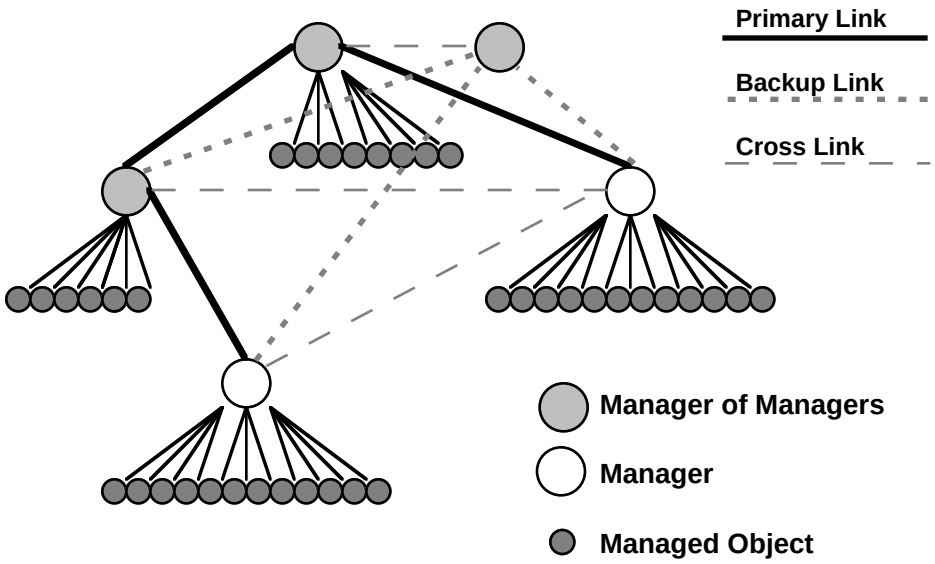


Figure 35. Compensated Hierarchical Architecture

Compensating the deficiencies of a hierarchical architecture to meet the ECS system needs would result in a certain decoupling of the strict hierarchical functionality, as shown in Figure 35. Provisioning to spatially replicate the highest level manager, or to allow direct coordination between lower level managers without the highest level manager's intervention quickly evolves the architecture to that of a management federation. In the example shown, the evolution to a federation of managers becomes clear with only three lower-level managers present.

5.2.2 Federated Architecture

A representation of a federated architecture is shown in Figure 36. Federation will be essential in large-scale distributed systems where the existence of sole ownership and control of resources cannot be assumed. As referenced by figure 35, the only way to achieve cooperation between autonomous systems without creating a hierarchical system is to use federation.

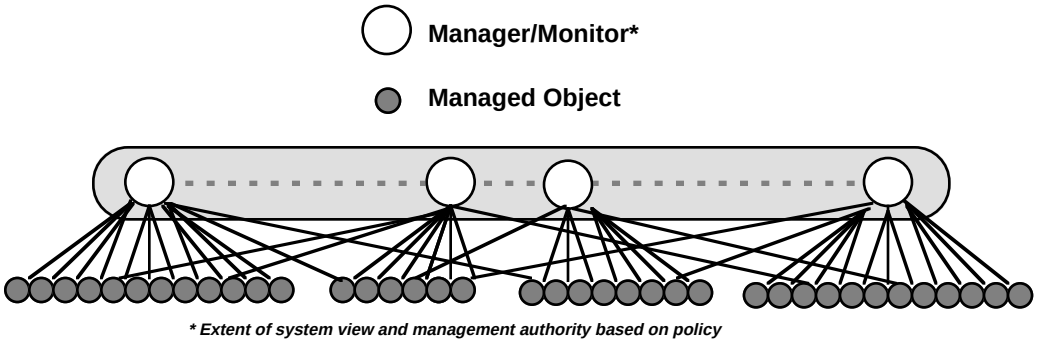


Figure 36. Federated Architecture

In a federated architecture, managed objects have the capability to interface to any number of managers, and not just its “parent” in the hierarchy. Unique domain managers have the capability to establish peer-to-peer relationships, eliminating the need for a single component at the top of the hierarchy. The actual setup is done in accordance with policy, potentially setup as memoranda of understanding between the management domains and implemented by the administrators of the respective domains. The advantage this approach brings to ECS is the capability for site-local personnel with domain expertise to interact directly with each other for problem resolution, rather than through a system-wide manager. Based on policy, this can be done with or without the system manager's direct involvement . The system manager would have the capability to monitor the situation, and step-in only when required.

Global namespace maintenance in a federated architecture is not required. This distributes namespace maintenance across the enterprise, improving response time for name service access and building in robustness.

The federated architecture emerging with enterprise management technology today allows for the integration of legacy management solutions. The approach is similar to that discussed in 3.3, where adapters encapsulate procedural interfaces. The ability to integrate legacy systems preserves existing investment and provides a smooth migration path to an effective distributed systems management architecture.

5.3 Implementation Architecture

5.3.1 Manager Architectural Alternatives

5.3.1.1 SNMP Model

SNMP was introduced in 1988 by the IETF (Internet Engineering Task Force) as its standard management protocol (RFC 1067). Although some implementations are being developed for non-TCP/IP transports (SNMPv2), SNMP is generally associated with TCP/IP transport. SNMP has achieved wide-spread acceptance and implementation because it is a protocol that is inherently simple: easy to implement with a minimal consumption of processor and network resources. The recommended transport for SNMP is the UDP (User Datagram Protocol), a connectionless transport which conserves network resources but is not a reliable transport. The Internet SNMP Management Model consists of:

- a managing entity (the Network Management Station or NMS)
- a managed entity (the SNMP Agent)
- a logical information store (the Management Information Base or MIB)

The SNMP agent is responsible for responding to management operation requests received from the NMS. The SNMP agent may also asynchronously send messages called traps to NMSs when the SNMP agent detects a device failure or other predefined event. SNMP is the management protocol used between the NMS and the SNMP agent. The SNMP model supports a management information base (MIB) for a logical store of information to support network management (RFC 1213).

5.3.1.1.1 Communications Model

5.3.1.1.1.1 SNMP Communications Model

SNMP consists of five message types or functions: GetRequest, GetNextRequest, SetRequest, GetResponse and Trap. SNMP management is based on polling the agents in a network for information. A variation of this basic mechanism is SNMP trap-directed polling, in which the NMS polls an SNMP agent in response to an asynchronously generated alarm or trap message. This type of polling is supported to relieve the network of the load of continuous polling. Trap messages are not delivered reliably however, so that it is unwise to eliminate polling entirely. The bulk transfer feature of SNMPv2, discussed next, may assist with reducing the impact of polling in SNMP management over a large network. Basic SNMP operations are shown in Figure 37.

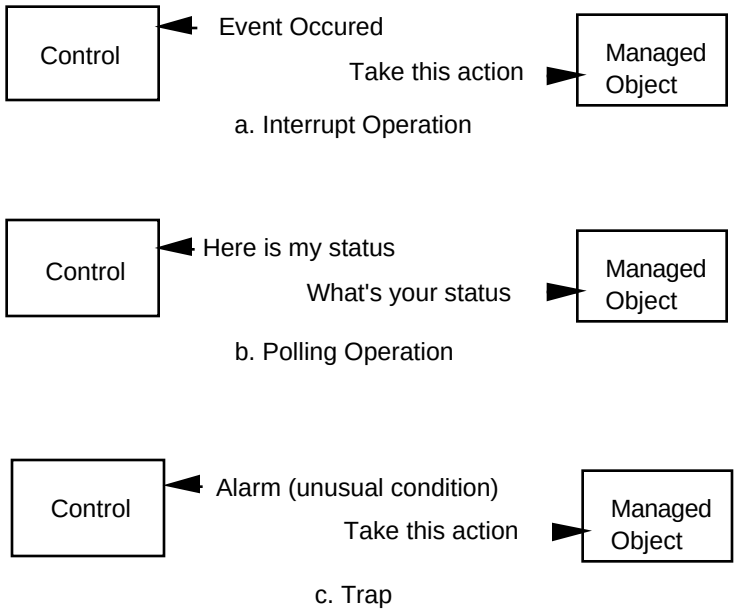


Figure 37. SNMP Operations Summary

5.3.1.1.1.2 SNMP Version 2 (SMNPv2)

The simplicity of the original SNMP (SNMPv1) also left some issues unaddressed, especially as networks increased in size and complexity. SNMPv2 addressed some of these criticisms:

- SNMPv2 supports bulk data retrieval in addition to the direct device polling supported in the earlier version. In very large and complex networks, such as ECS, this reduces load that could adversely affect the overall state of the network.

- SNMPv1 was oriented toward a centralized approach to management. SNMPv2 supports manager-to-manager communications for better distributed management of multiple network domains.
- SNMPv2 supports security enhancements, including controlled access to system and device agents and DES encryption. (RFCs 1351, 1352, 1353)

Vendors are beginning to support some of the SNMPv2 features. For a distributed systems management implementation SNMP, ECS should seek to implement as many of the robust version 2 features as they become available and are widely supported.

5.3.1.1.2 Information Model

5.3.1.1.2.1 SMNP MIB I and MIB II

The SNMP MIB (RFC 1066:1988) was originally designed to identify enough useful information without impacting network resources. This subsequently became known as MIB I. It contained 114 pieces of information that were thought to be appropriate to managing network resources.

The relatively straightforward MIB I structure (static database with tree structure) is adequate for many networks devices and resources. It has some potential difficulties in handling composite device structures, where different components may require their own database models that cannot be unified into a single MIB due to its static structure.

SNMPv2 resolves some of the above limitations. It can collect sets of data and it places intelligence "in the network". MIB II (RFC 1213) contains 170 information items, organized into 9 groups: system, interfaces, ip, icmp, tcp, udp, egp, transmission and snmp. The MIB structure supports unique organizational and vendor-specific MIBs or MIB extensions for additional product-related management functions.

5.3.1.1.2.2 RMON-MIB

RMON-MIB is a proposed IETF standard. It provides improvements in intelligent monitoring, especially over large network implementations. It requires less network traffic and overhead than SNMP proxy agents. It collects network information from remote devices (segments) and supports nine attribute classes including:

- Statistics (fragments/collisions),
- History (stats over time)
- Alarms (compare current stats to thresholds)
- Hosts (information on active hosts)
- HostTopN (information on highest host in particular stat)
- Matrix (traffic between nodes)
- Filter (packet filtering by equation)
- Packet Capture (provides packet capture)

- Event (controls events form a device)

An extension was also added to the RMON MIB (RFC 1513) for Token Ring support, which is called Token Ring for events related to this topology. Extensions are being considered and working groups are being formed for a number of other transport media, including ATM.

5.3.1.1.3 Applications Model

Numerous vendor applications that interface with the SNMP model are available today for network management. The products work well for their intended purpose - to monitor and manage networks. Some products are now emerging that use XMP, an X/Open branded API for support to SNMP and CMIP/CMIS protocols. Vendor-specific GUI's are a major impediment at this time in promoting cross-platform interoperability of vendor and third-party application products.

5.3.1.1.4 SNMP Strengths and Weaknesses

Because of its simplicity and ease of implementation, SNMP has gained and is continuing to gain wide industry support, far greater than any other management protocol. SNMP's native transport, TCP/IP is also gaining wide industry support. In addition, major industry players, such as Novell, are supporting SNMP on other transports, such as IPX. Many consider that the standards body for SNMP, the IETF, is more efficient in delivering appropriate standards in a consistent and timely manner, that result in fairly rapid product implementations than organizations such as the ISO.

In spite of these pluses, networks and computing environments have become increasingly complex and distributed, and SNMP, while efficient in many areas and still evolving, has limitations in expansion toward systems management. These limitations will become more apparent through the other architectures to be discussed.

5.3.1.2 ISO/CCITT Model

The Network Management extensions to the ISO model began in the early 80's. The initial design phase of these extensions was not completed until nearly a decade later. (ISO/IEC 7498-4:1989). Unlike the IETF's more pragmatic approach to identifying and solving the most critical issues, the ISO network management model approached the problem very globally and very generally. The OSI management components include:

- A managing system, taking the manager role, similar to an SMNP Network Management Station.
- A management communication protocol, Common Management Information Protocol (CMIP)
- Management Information (Management Information Tree (MIT) database)

The ISO management model provides, shown in Figure 38, provides more management flexibility and is a more complete management structure for systems management than SNMP. Five Systems Management Functional Areas (SMFAs) are supported by the ISO model, including fault, performance, configuration, security and accounting. The tasks associated with each of these areas were also identified. It became apparent that some tasks were related to more than one SMFA and

the more recent OSI specifications, such as the System Management Overview (ISO/IEC 10040:1991) are evolving to a framework structure.

The ISO framework introduced the concept of administrative management domains, similar to administrative domains found in X.400. Real systems are organized into sets to meet certain requirements, such as fault management, or security management; to assign the roles of agent and manager; and to establish control over the domain. The administrative domains are called functional administrative domains by ISO.

Appendix D provides reference additional notes on the ISO/CCITT management model.

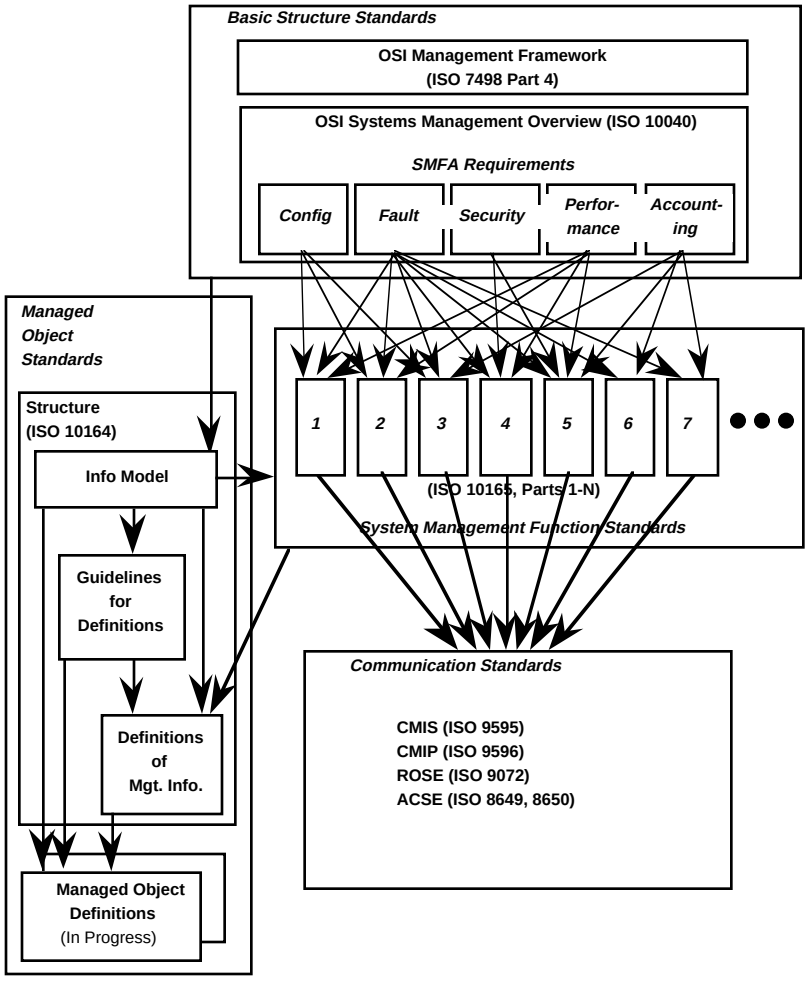


Figure 38. ISO Systems Management Framework

5.3.1.2.1 Communications Model

OSI CMIP is a connection-oriented management protocol to ensure reliable delivery. CMIP utilizes a more complex informational model than SNMP. OSI CMIS (Common Management Information Service) provides the management services including the following:

M-Create: Creates an MO instance record in the MIT

M-Delete: Deletes an MO instance from MIT

M-Get: Retrieve information. Aggregate (bulk) and selective (filtered) retrieval

M-Cancel-Get: Cancel retrievals

M-Set: Change an attribute value

M-Action: Invoke an MO operation

M-Event-Report: Generate an MO event report to a manager

These services extend management capabilities beyond that of the SNMP model. While there are comparable SNMP commands to some of these CMIS Services, there are no comparable SNMP commands for M-CREATE, M-DELETE, and M-CANCEL-GET. This has made mapping between the two management protocols difficult. CMISE uses ISO ACSE and ROSE to support services.

CMOT is CMIP over TCP. GDMO is well received in the Internet community and the use of CMIP over the Internet reliable transport is required in order to have ISO-based systems management of Internet-based resources.

5.3.1.2.2 Information Model

The ISO Management Information Model (MIM) (ISO/IEC 10165-1:1991) is based on object oriented technologies and techniques and is a highly flexible model. This model manages resources, such as systems, protocol layer entities, and devices, as managed object classes. Each managed object class may include the following properties: attributes, notifications, operations and behavior. The OBJECT_IDENTIFIER, that functions as the source/target of all CMIP data units, is comprised of a single unit of encapsulated, self-contained properties.

The object-oriented techniques used in this model, such as inheritance (subclasses) and polymorphism, support flexibility far beyond simple procedural models. Object Class definitions are documented using the format of Guidelines for the Definitions of Managed Objects (GDMO ISO/IEC 10165-4:1991). The GDMO contains a number of templates that are used to represent specific managed objects classes and their properties. The GDMO templates define the representation of the object classes, while its properties are represented in CMIP data units.

5.3.1.2.3 Applications Model

The five functional areas defined by ISO for systems management are fault, configuration, accounting, performance, and security management. *Fault Management* is used to detect, isolate, and repair problems. It encompasses activities such as the ability to trace faults through the system,

to carry out diagnostics, and to act upon the detection of errors in order to correct the faults. It is also concerned with the use and management of error logs. Fault management also defines how to trace errors through the log and time stamping of the fault management message.

Accounting Management is needed in any type of shared resource environment. It defines how network usage, charges, and costs are to be identified in the OSI environment. It allows users and managers to place limits on usage and to negotiate additional resources.

Configuration Management is used to identify and control managed objects. It defines the procedures for initializing, operating, and closing down the managed objects, and the procedures for reconfiguring the managed objects. It is also used to associate names with managed objects and to setup parameters for the objects. Lastly, it collects data about the operations in the open system in order to recognize a change in the state of the system.

Security Management is concerned with protecting the managed objects. It provides the rule for authentication procedures, the maintenance of access control routines, the support of the management of keys for encryption, the maintenance of authorizations facilities, and the maintenance of security logs. Security management is in the formative stages, but it is certain that the standard will really extensively on the Directory Services Standards for security support.

Performance Management supports the gathering of statistical data and applies the data to various analysis routines to measure the performance of the system. It permits analysis routines to measure the performance of the system. It permits the use of models to determine if a system is meeting the required throughput, providing adequate response time, approaching overload, and/or if a system is being used efficiently.

Underlying all of the functional areas are functional standards to support the functional areas. The current list of functional standards include:

- Object Management Function (ISO DIS 10164-1, IS in '91; CCITT X.730)
- State Management Function (ISO DIS 10164-2, IS in '91; CCITT X.731)
- Attributes for Representing Relationships (ISO DIS 10164-3, IS in '91; CCITT X.732)
- Alarm Reporting Function (ISO DIS 10164-4, IS in '91; CCITT X.733)
- Event Report Management Function (ISO DIS 10164-5, IS in '91; CCITT X.734)
- Log Control Function (ISO DIS 10164-6, IS in '91; CCITT X.735)
- Security Alarm Reporting Function (ISO DIS 10164-7, IS in '91; CCITT X.736)
- Security Audit Trail Function (ISO CD 10164-8, IS in '92; CCITT X.740)
- Objects and Attributes for Access Control (ISO CD 10164-9, IS in '93; CCITT X.741)
- Accounting Meter Function (ISO CD 10164-10, IS in '92; CCITT X.742)
- Workload Monitoring Function (ISO CD 10164-11, IS in '92; CCITT X.739)
- Test Management Function (ISO CD 10164-12, IS in '92; CCITT X.745)
- Measurement Summarization Function (ISO CD 10164-13, IS in '92,; CCITT X.738)

- Confidence and Diagnostic Test Classes (ISO WD 10164-y; CCITT X.737)
- Time Management Function (ISO WD 10164-z, IS in '93; CCITT X.743)
- Software Management Function (10164-q, IS in '94; CCITT X.744)
- Performance Management Function (CCITT X.746)

The ISO architecture is object-oriented, and the functional standards are similar in scope to, and overlap, the object services and common facilities in process of definition by OMG. Management application vendors would seem to be backing OMG object services technology more than the ISO system management functions. X/Open has recently branded Tivoli's Management Environment and is in the process of refining management services that complement the set of OMG object services. Additional discussion of this is found in the discussion of X/Open. Additional discussion of ISO/CCITT object model reconciliation with the OMG object model is provided under the Network Management Forum and Object Model reconciliation sections of 5.3. For detailed information of the system management functions, the reader is directed to the ISO 10164 series.

5.3.1.3 GNMP

The Government Network Management Profile (GNMP) is a NIST document the adopts the ISO/CCITT CMIP/CMISE protocol and services to integrate managers. CMIP/CMISE from managers to managed objects is not specified under GNMP - only the interface between managers. The first seven system management functions (ISO 10164-1 through -7) are specified in the document. The focus of GNMP is to build a hierarchical network management system, and only addresses interoperability between network management components. The specification takes a good look at classification of managed objects, but stops short at the network level. NIST is involved working with the NMF (see discussion below) in being an integral part with OMNI*Point* 1 Specifications. Reference Notes on GNMP are provided in Appendix E.

5.3.1.4 OSF DME

The OSF Distributed Management Environment is an architecture with many distinct parts. The basic environment, pictured in Figure 39, is comprised of Distributed Services, a Network Management Option (NMO), and a Management Framework (MF). The MF is a component of DME that has seen quite a bit of upheaval and duress. Originally, the MF was based on technology from Tivoli that utilized an object framework incompatible with OMG CORBA. When it became clear to OSF that the future of distributed systems management would be based on OMG CORBA, OSF was faced with the challenge of making the MF CORBA compliant. After missing a few deadlines, Tivoli dropped out of OSF and vowed to make their Tivoli Management Environment (TME) CORBA compliant. Development at OSF continued, to the present framework today. As can be expected, change is constant in the industry. Recently, OSF announced plans to drop the MF component of DME. The event services of DS will be included with release 1.1 of OSF DCE, due in September 94. The rest of the DS complement the communications sub-architecture and will be considered as potential design implementations for some of the OMG object services (e.g., software licensing, software distribution). The NMO is largely available today as part of the X/Open branded XMP API for SNMP and CMIP/CMIS access, bundled with HP, DEC and IBM

network management products. Reference material of DME and its components is provided in Appendix F.

A major announcement was made at UniForum (April 94) of the future direction of OSF. OSF is recently talking with COSE; it is believed the forthcoming announcement will help to better align the major consortia to aid in the streamlining of product development through to standards. Under the anticipated scenario, COSE would identify technology problems and solicit the industry for a solution. X/Open would brand the solution, and OSF would be the integrator/implementor. ECS views this as a positive move on behalf of consortia in creating greater bottom-up market push of advanced technology to realizable product that can later be standardized. HAIS is following this activity closely.

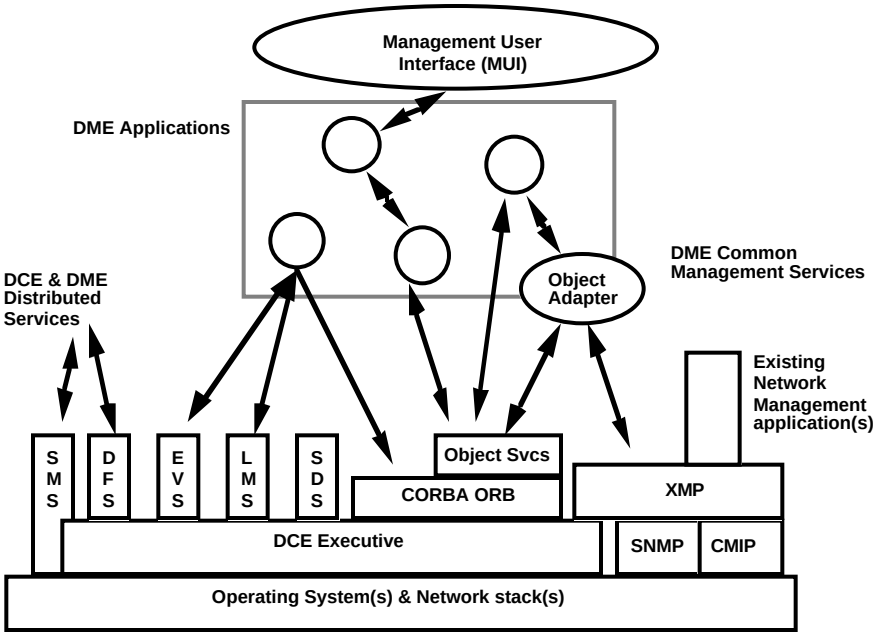


Figure 39. DME Functional Architecture

There are many benefits of DME that ECS has already drawn benefit of. The use of DS in DCE and implementation for OMG object services saves development. The NMO XMP API, brought about by OSF, is a major first step in integrating legacy SNMP and CMIP/CMIS network management systems. Finally, the DME work has provided much insight to the integration of systems management into a CORBA and OMG object services architecture, and prompted much work within the NMF on the reconciliation of ISO/CCITT object models and the further definition of managed objects. Another major benefit of the DME effort is the continuing development of a management GUI (MUI) based on Motif. The MUI will be in a future enhancement of OSF's Motif, and will enable portable third party management applications between multiple vendor system management solutions that adhere to the MUI.

The framework architecture of the MF is shown in Figure 40. The planned components supplied with the MF included the management services and the object services - the MUI is part of Motif. Management applications, fully portable by the adherence to standardized interfaces, are developed by ISV's, end users, and system vendors. DCE is shown as the communication mechanism, but is not required - the architecture is on top of CORBA. Many ORB implementations can be made available that the MF would integrate directly onto, via the OMG object services. In the case of the figure shown, the DCE services are abstracted into the OMG object services. Alternative implementations would implement OMG object services in different ways, and interoperability of the MF would not be affected.

The specific object services (ref. section 3.3) that are used for the MF include lifecycle, synchronization, persistence, event notification, security, naming, message catalog, and time services. The message catalog service, catalogs internationalized messages in a distributed fashion. Identified management services that would be developed by OSF include the management user interface, maps and collections, discovery, monitoring, domains/policy, adapter objects, and scheduling services. The management services facilitate migration of existing legacy (network management) applications into the MF.

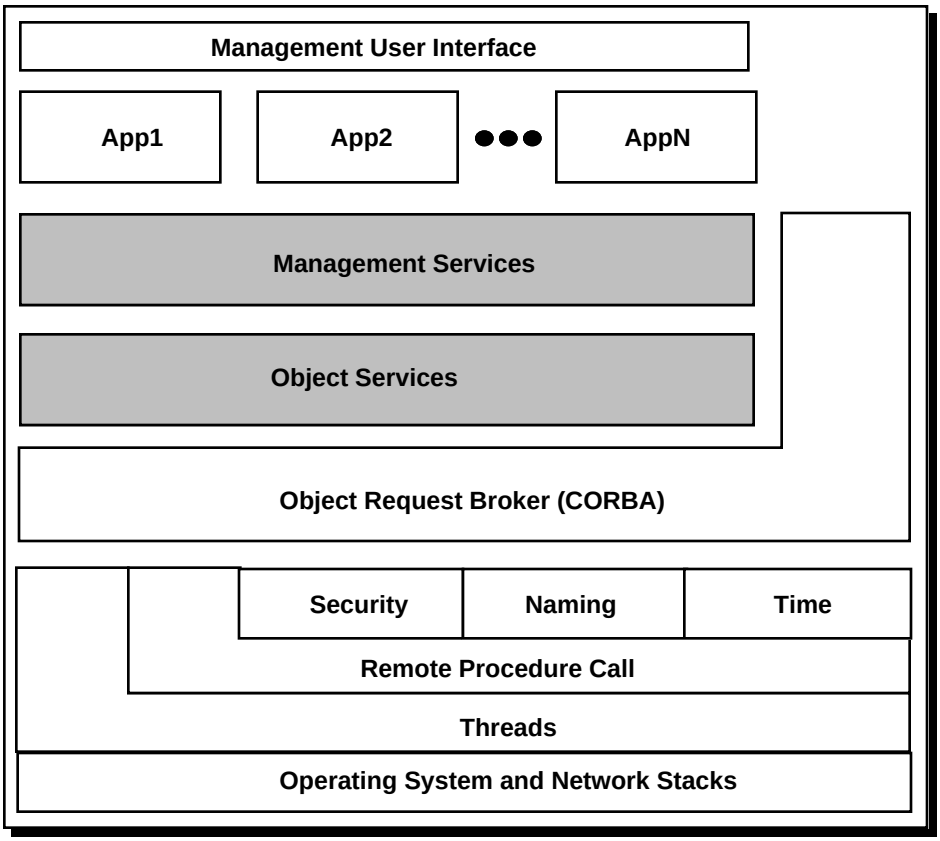


Figure 40. DME MF Architecture with DCE Context

The management user interface, previously discussed, provides tools and services needed to build user interfaces for distributed OO applications. Important features of the MUI are the ability to present maps, to support dialog interaction, and to separate presentation from interaction.

The maps and collections service provides uniform management of the relationships between managed objects and are key to the integration of network and systems management. Maps and collections frequently act as an applications primary presentation vehicle.

The discovery service provides the basis for managed resource discovery (such as new systems, devices, etc.), and is required for NMO integration. The monitoring service provides for the detection of state changes in managed objects, alarms, etc, and is also important for NMO integration.

The policy service provides for the creation and management of domain-specific policy for managed objects (such as how to treat all users in a group, all software depots of a certain class, etc.). Adaptor objects, discussed in section 3.3, interface to the procedural interface of NMO to provide OO interfaces to more traditional services. A final service, scheduling, provides a means to schedule periodic and routine tasks.

The UniForum announcement should provide more light on the future direction of OSF, and which consortia will pick up the work of DME. COSE has a systems management working group that will likely take over the effort.

5.3.1.5 OMNIPOINT

The OMNI*Point* 1 management model is associated with the Network Management Forum (NMF). OMNIPoint stands for Open Management Interoperability Point. OMNI*Point* 1 are a set of user-specified implementation requirements, standards, testing methods and tools, and object libraries providing an articulate path toward integrated, automated network management. The mission of NMF can be described as one to find solutions providing widespread interoperability of management information in order that the service needs of users of systems can be managed effectively. The OMNI*Point* program is supported by NMF, OMG, X/Open, OSF, UI, SPAG, NIST, the OSE Workshops, and others.

The Network Management Forum (NMF) is an industry consortium which has compiled a multi-volume set of standards, specifications and technologies for achieving open management interoperability in the purchase or development of management applications. The NMF is working directly with an OSF Special Interest Group (SIG) and the Object Management Group (OMG) on object standards and interoperability to support heterogeneous enterprise management. The NMF's OMNI*Point* 1 is a compilation of standards rather than a reference technology or an true architecture. Many DME technologies and specifications are also included in the OMNI*Point* 1 specifications.

One of the critical areas that OMNI*Point* is focusing on is to specify mechanisms to provide interworking of OSI, OSF/DME MF, and OMG CORBA object models, and providing translation tools between GDMO and IDL and the DME I4DL (Note: the I4DL conversion has probably been dropped). Also, OMNI*Point* is working with the Internet community to produce RFCs for placing the structure for management information (SMI) from MIB II into GDMO format.

As can be seen in Figure 41, the OMNI*Point* model is similar to OSF DME architecture in many aspects. Management applications achieve portability by a common interface at the user and communications interfaces. Management services, working with OMG CORBA provide adaptation to the legacy environment. Legacy SNMP and CMIP/CMIS applications are integrated through the use of the X/Open XMP API. OMNI*Point* actually goes further than the DME architecture by examining integration strategies of the management information model in addition to the communication services.

OMNI*Point* acknowledges many object-oriented approaches to managing networks have been, and will be used. The OMNI*Point* model identifies components for an integration framework for users and developers of network management systems. The focus of OMNI*Point* is based on the OSI/CCITT object model, but recognition is made to the need to interoperate with SNMP and CORBA paradigms of network and system management. These object paradigms are continuing to be analyzed by NMF and the OMNI*Point* partners to start the specification of tools and algorithms to map between the paradigms. A draft document comparing the OSI and OMG object models has been developed in part due to OMNIPoint activities. Future OMNI*Point* work plans include:

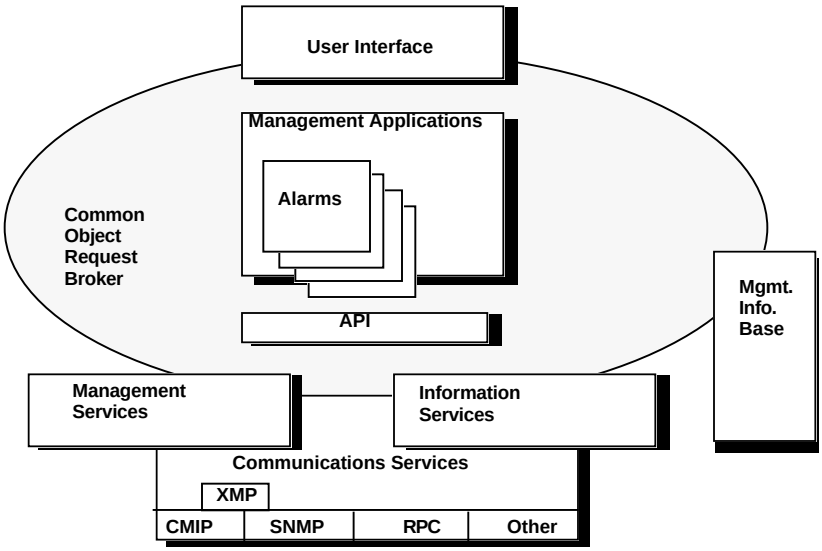


Figure 41. NMF OMNIPoint Model

- Mechanisms to permit the interworking of OSI based systems with others based on specifications such as OSF/DME and OMG/CORBA
- Notation translation algorithms and tools needed to translate between GDMO, CORBA IDL, and DME/I4DL notation techniques

OMNI*Point* additionally recognizes the need to work with importing SNMP-based management information, and has been actively mapping OSI and SNMP MIB with Internet participants. Draft deliverables (currently available as Internet draft RFCs) have been produced describing:

- Internet SMI MIB II in GDMO format

- Internet SMI Party MIB in GDMO format
- Mapping algorithm from SMI to GDMO
- Proxy Agent specification

5.3.1.6 ODP and OMG

The RM-ODP aspects of system management are at the atomic level. The management issues covered include object lifecycle, resource management and policy management for individual objects. All RM-ODP objects include a management interface. Objects inherit these supporting mechanisms to reduce programming burden.

An object performs a management function by invoking an operation on its management interface. The stages in the life of an object include:

- creation - resources are allocated for object template
- service offer - reference to interface provided to trader
- migration - to balance load, reduce latency
- checkpointing - for recovery after host failure
- passivation - when idle to disk, reactivate when invoked
- service offer withdrawel - from trader
- termination - release resources

OMG provides these management functions in the following way:

- creation - through lifecycle service and factory objects
- service offer - inherent to trading service
- migration - inherent to lifecycle service
- checkpointing - inherent to transaction service
- passivation - programmer defined
- service offer withdrawel - inherent to trader
- termination - inherent to lifecycle service

Management objects are not presently envisioned in the OMG framework. Groups involved in these issues include X/Open, NMF and the OSF MAN SIG. For the ODP vision of system management to be integrated within advanced systems management architectures, management services providing support for policy management will be required.

5.3.1.7 X/Open

X/Open Company Ltd. was founded in 1984 as an international, independent organization dedicated to developing an open, multivendor applications environment. X/Open designed XPG (X/Open Portability Guide) as a vehicle for implementing open systems in the real world. XPG is

an evolving portfolio of application programming interfaces (APIs), protocols, and other specifications that are supported with an extensive set of conformance tests. A distinct X/Open trademark is carried only on those products that comply with X/Open portability definitions. The latest release, XPG4, includes specifications covering interoperability and communications.

The X/Open's technical working groups draw on several sources--users, standards bodies, suppliers, and various consortia--to determine the specifications, ensure that they are aligned with relevant formal standards. When specifications are developed in advance of commercial implementations, they are first published as preliminary specifications. Only when the specifications have been tested and shown to be fully practical are they published as full X/Open specifications and used as the basis for branding.

X/Open recently adopted Tivoli Systems technology as the foundation for distributed systems management. The technology solution conforms with OMG/CORBA. The technology submitted by Tivoli provides a stable link to a systems management framework for systems vendors and application developers. A framework specification, which will evolve over the next year, is publicly available. System administration applications based on the interface specification are already being fielded. The X/Open branded architecture for distributed systems management is shown is Figure 42.

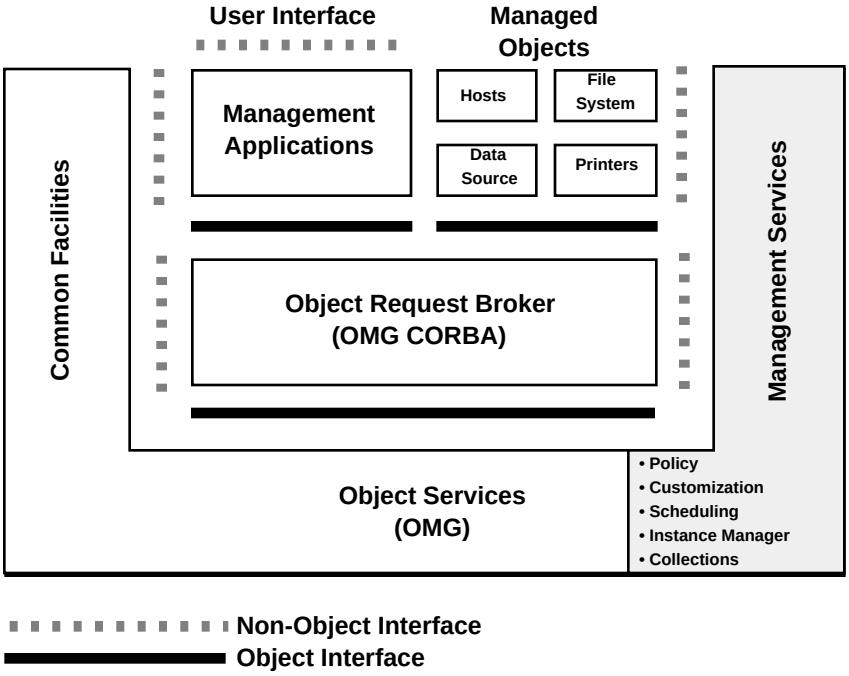


Figure 42. X/Open Systems Management Framework

The X/Open branded architecture is built on the OMG CORBA and Object Services, and specifies an approach to the development of open system administration applications. A set of management services are defined that allow management specific interfaces to be common across environments,

allowing the development of heterogeneous, interoperable applications. The specific object services deemed required for systems management by X/Open include lifecycle, event, naming, relationship, transaction, concurrency, and security services. Management services include the following:

- *Collection Service:* this service organizes objects into groups, providing the aggregation of many-to-many relationships
- *Policy-Driven Base Service:* this service allows common operations to be performed on all policy-driven objects within a system administration application. The common operations allow objects to be managed, grouped, and named.
- *Instance Management Service:* this service provides the structure necessary for designing objects to support system administration applications by encapsulating a series of client operations.
- *Policy Management Service:* this service gives administrators a way to customize applications to their specific needs.
- *Scheduling Management Service:* this service invokes operations at an administrator-defined time, and is essential for the scheduling of periodic tasks
- *Customization Management Service:* this service allows a system administrator to customize the behavior of their applications to suite organizational-specific requirements

The management services are key to building an application foundation so that systems management applications and objects can be defined and built to span multiple vendor framework implementations. The specification does not address the graphical user interface, which is being addressed by OSF Motif's MUI, or application-specific resource interfaces (the applications objects defined in section 3.3). Additionally, the specification does not discuss information model aspects of systems management (work in progress with ISO GDMO, NMF, and OSF MAN SIG).

5.3.2 Managed Object Alternatives

5.3.2.1 Managed Object Working Group Activities

There is considerable activity in the definition of managed objects. The major consortia outlined in section 5.3.1 are working in this area, either directly or indirectly. The activities of several of these workgroups are outlined in this section.

5.3.2.1.1 IETF Host Resources MIB (HostMIB)

The IETF Host Resources MIB Working Group defines a UniForum MIB for objects useful in the management of hosts. The source MIB can be translated into GDMO according to ISO/CCITT Internet Management Coexistence (IIMC) translation rules. Work is in progress, including work with the DSIS and DMTF groups outlined below to support SNMP management of (PC) desktop objects defined by these organizations, described below.

5.3.2.1.2 DSIS

Distributed Support Information Standards Group has a requirements specification that identifies data necessary for organizations performing service, support, and management activities for networked systems. The DSIS is developing object definitions for desktop devices and subsystems to populate a management information file that works like a Management Information Base (MIB) to detail pointers to relevant information. It is expected that the DMTF API will access information defined by the DSIS group. These definitions will include the name, vendor, model, and location of desktop components, including NICs, modems, drives, and video gear for PCs.

5.3.2.1.3 DMTF

The Desktop Management Task Force has work ongoing in defining related common management definitions in the Management Information Format (DMI MIF). It is expected that the DMTF will address the object definitions specified by DSIS. DMTF is working with the IETF Host Resources MIB (HostMIB) working group to ensure desktop objects will be manageable under SNMP. This may be provided by an API and or gateway implementation. The work has support for PC LAN devices and desktops from a number of major vendors. Submission to or support for major management standards have not been announced, although some support vendors believe SNMP support is critical to acceptance.

5.3.2.1.4 IEEE

Work is on-going in POSIX operating system managed object definitions and storage related managed objects in Mass Storage Reference Model. The current work on POSIX 1003.7 (recently renamed POSIX 1387) (System Administration areas, such as user-account management, software distribution, devices, file systems, processes and backups, etc.) assumes that a framework will be available to provide common services across platforms. The committee does intend to address an API to this framework, but expects the framework to be provided by organizations such as OSF, Tivoli, ISO and other standards organizations. It is expected that the interface will support the managed object definition standards supported by these organizations, which have been described in previous sections.

5.3.2.1.5 Unix International

Unix International (UI) was working the definition of performance and configuration metrics for a Performance Measurement Data Pool requirements definition. UI disbanded in December, it is not known if this work was transferred to another organization.

5.3.2.1.6 OSF Management SIG

The Hilton Head Object Group (HHOG) is a forum in which the OSF Management Special Interest Group (SIG) can test its current work on a standard object model. The OSF Management SIG has various task forces trying to define common object definitions in the areas of operating systems, license management, network events, and even the management of DME itself. The SIG is using as the basis for this work the International Standards Organization's (ISO's) Guidelines for the Definition of Managed Objects (GDMO) language, which is backed by the NM Forum. The GDMO definitions are intended primarily as a reference specification that can be used to create

mappings in more widely used languages, particularly the OMG's Interface Definition Language (IDL). X/Open's anticipated role will be to manage the specification.

5.3.2.1.7 OMNIPOINT

As previously discussed, OMNI*Point* future work plans include:

- Notation translation algorithms and tools needed to translate between GDMO, CORBA IDL, and DME/I4DL notation techniques.

Draft deliverables (currently available as Internet draft RFCs) have been produced by OMNI*Point* describing:

- Internet SMI MIB II in GDMO format
- Internet SMI Party MIB in GDMO format
- Mapping algorithm from SMI to GDMO
- Proxy Agent specification

OMNI*Point* is additionally supporting work in the integration of the OSI/CCITT and OMG object models.

5.3.2.2 Relationships of Managed Object Working Groups

Table 5 presents a tabular summary of the current work in progress within the managed object working groups of several prominent organizations involved with system management.

Table 5. Managed Object Definitions in Work

	Host MIB	DSIS	DMTF	IEEE	UI	OSF/MS	NMF	ISO
Managed System	N	N	Y	N	Y	N	Y	Y
Hardware	(See GDMO Hardware Managed Object Specification)							
Physical Devices	Y	Y	Y	N	N	N	?	N
Operating System	Y	Y	Y	N	Y	Y	?	N
Logical Devices	N	N	Y?	Y	Y?	N	N	N
Processes	Y	Y	Y	N	Y	Y	N	N
File System	Y	Y	Y	N	Y?	Y	N	N
Software	Y	Y	Y	Y	N	N	?	Y
Application	N	N	N	N	N	Y	N	N
Queue	N	N	N	N	N	Y	N	N
Queue Entry	N	N	N	Y	N	Y	N	N
Users	Y	N	N	Y	N	Y	N	N

5.3.3 Systems Management Sub-Architecture Discussion

5.3.3.1 Object Model Reconciliation of OMG and ISO/CCITT

The NMF took on with OMG the task of comparing the OMG and ISO/CCITT object models. The promising architectures discussed in this paper all are looking to CORBA and the OMG object services as the advanced infrastructure on which to build a federated, large scale distributed systems management system. The managed objects, however, are almost all being defined in GDMO. A method to map GDMO objects onto the OMG IDL is therefore required. This section compares the object models developed by OMG for object oriented software development and ISO/CCITT for OSI network management. It identifies areas of conflict in the two models and provides alternatives and strategies to reconcile differences. Detailed information of the object model comparison can be found in the February 1993 Report of the Joint NM Forum/OMG Taskforce on Object Modeling entitled "Comparision of the OMG and ISO/CCITT Object Models".

According to the report, there is an high degree of agreement between the basic aspects of the two models. The fundamental notions of objects, object taxonomy, attributes, operations, state, behavior, and encapsulation are virtually identical. The conclusion is very important for the realization of OSI-conformant network management products using OO-software development systems, like CORBA. Because the models are so close, less actual translation code will be required to bring GDMO-defined managed objects and applications into CORBA, improving accuracy and robustness of implementation.

There are some significant differences, however. The models differ in three significant aspects; the support for events, multiple replies, and late binding. A number of less important aspects differ as well, such as specification techniques and associative references, but these are not major issues. In two ways, the differences in the two models actually are reported to complement each other - in interface type and intended use. The fundamental differences will require reconciliation in order for ISO/CCITT and OMG object models to interwork.

The report discusses three approaches for reconciliation. These are model alignment, run-time mediation between implementations of the models, and notational mapping tools. Notational mapping tools is the current path selected by NMF in its workplans with OMNI*Point.* Total model alignment is impractical given the depth of backing and years of effort each model has behind it, but the development of flexible profiles to minimize the differences is highly recommended (and currently in work). Run-time mediation would require the development of extra software to match differences in the specification (the ISO/CCITT model), and the implementation (OMG model). This could happen in the future. Notation translation tools involves syntax checkers, data-structure generators and ASN.1 compilers for semi-automatic translation between the two models. Fully automated translation is never expected.

5.3.3.2 Framework Architecture

Five major distributed systems management architectures were presented that have the potential to provide to ECS a truly distributed and federated systems management solution. The primary features of each architecture are discussed below.

The DME architecture, while no longer in development, offered many good concepts on the integration of legacy management applications. This was reflected in the selection of the management services, including the discovery, monitoring, maps/collections, and adaptors. The MUI is a very important concept to effect maximum portability of management applications, both now and for the future. The recommended object services matched those of X/Open, and further suggested that time and persistence services would be beneficial to management services. The focus of DME was to provide a total Open Management Environment - a shared goal at COSE. This may not be possible even for quite a long time. The X/Open specification for an OO system administration environment does not discuss integration with the legacy environment and the final X/Open specification is not due for another year.

The OMNI*Point* framework is a very useful architecture through the inclusion of all aspects of distributed systems management - the communications model, information model, and management application model. This and the ISO model shared many features, integrated on a commercially viable communication infrastructure - the OMG CORBA. The efforts in NMF for unification/reconciliation of GDMO with CORBA IDL and IETF is to be commended.

The ISO framework is relevant to ECS largely in the information model, but the communication model and application framework at the system management functions level (ISO 10164-1 through -n) will likely never happen. While FCAPS (fault, configuration, accounting, performance, security) is a useful classification of the system management responsibility space, alternative high level classifications are appearing based on a bottom-up product development integrated on OMG object services, consortia common facilities, and X/Open and DME management services. The work of OSI in response to this will need to be re-examined and NMF responses will need to be made known. The classification of managed objects using the GDMO templates should continue to be the predominant foundation to a system management information model.

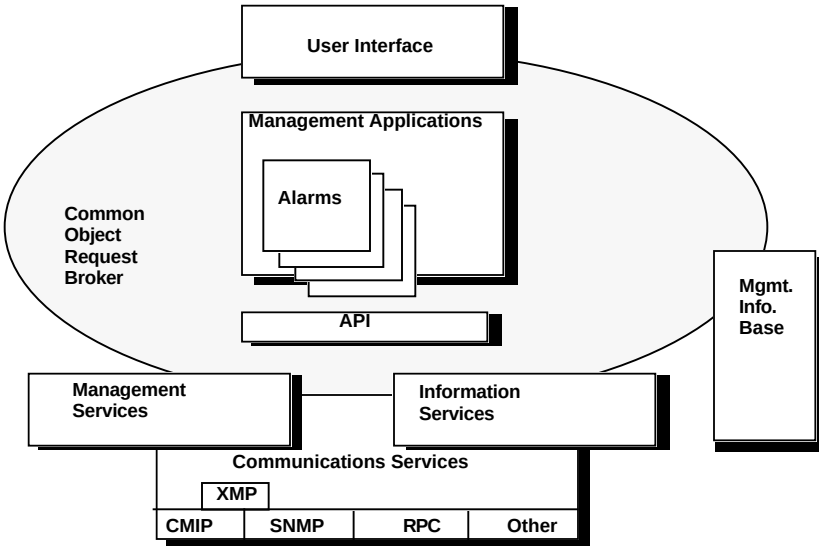


Figure 43. NMF OMNIPoint Framework Architecture

Based on the UniForum announcement, the consortia may be much better aligned to serve end-users by identifying promising technologies and holding fast to it straight through to the standards committees. If X/Open is to be truly recognized as the official branding body, then it's endorsement of Tivoli's CORBA-compliant management environment is a very strong sign of the management services on which to base OO system management applications. The X/Open management services for policy and customization take a refreshing look at a administrator-centric approach to management that could well prove to be the correct approach. What is lacking in the X/Open specification is compensated for by the other architectures - MUI from OSF Motif, Information Model from NMF and ISO, and communications model of OSF DME. The overall integrating architecture of the NMF accommodates the 'best-of-breed' from each of these architectures and is recommended as the ECS framework (implementation) architecture for systems management. It is shown again here in Figure 43.

5.3.3.3 Aspects of Systems Administration

5.3.3.3.1 System Administration Service

Figure 44 is provided to show how a main user interface for a system administrator might appear in the distributed systems management architecture. The functional requirements (fault, configuration, etc.) appear as applications available to the administrator. Collections or single objects could be queried, controlled or acted on by the administrator applications. Each of these applications would be based on an object-oriented database of objects developed with System Management Infrastructure objects. COTs databases that comply with the Query and Persistence Services would be used.

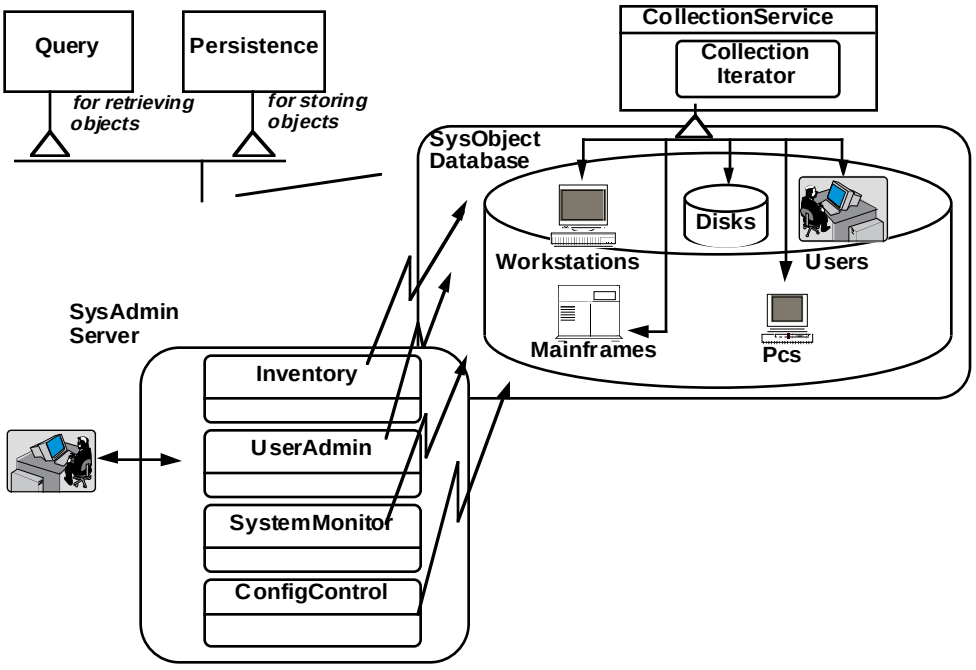


Figure 44. System Administration Service

5.3.3.3.2 Managed Objects in the OMG Paradigm

The management services branded by X/Open provide the fundamentals for logically modelling or physically storing objects for management purposes. These services define the operations that all managed objects can inherit and implement. Managed objects provide a model of the physical resources that are defined within the bounds of a system, and are subject to the policy of the region to which they are a part.

A region is comprised of managers and managed objects controlled by policy objects. One of the many resources in a region is a node. A node is a manager object comprised of possibly many capsules, created by factory objects. A capsule contains a nucleus object which provides uniform access to facilities and resources (such as processing, memory and communications) provided by the underlying operating system (see Appendix A for more detail).

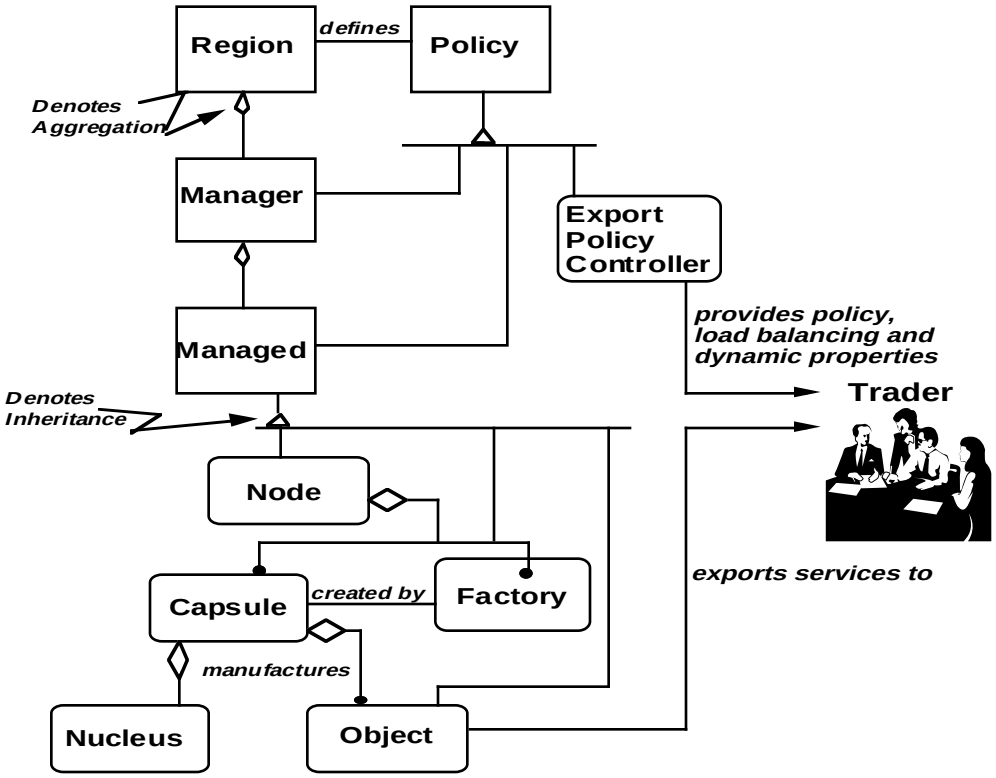


Figure 45. Management Infrastructure Objects

Managed objects export their services to a Trader object to register them in a systems offer space. ExportPolicyController objects interact with a Trader to enforce a region's policies whenever local services are provided. The associations of the manager, managed object, and trader are shown in Figure 45.

Policy may used to control the general activities of objects in a region. Figure 46 is an example of how local policy might direct an unused service to passivate to disk to save system resources.

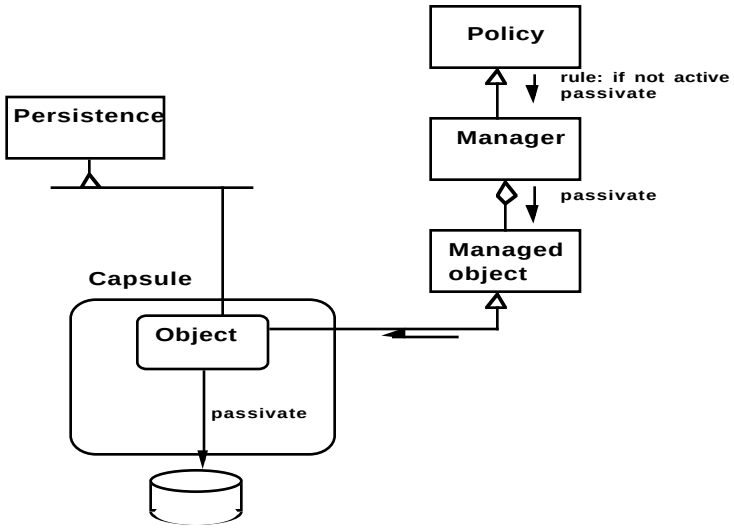


Figure 46. Example of Policy to Conserve Resources

5.3.3.4 Management Application Classes

System Management refers to the configuration, control, and change of computing hardware, operating systems, and software applications. The following list identifies many of the functional requirements for the domain of Systems Management:

- Configuration Management
- Performance Management
- Accounting Management
- Software Management
- Fault Resolution Management
- Security Management
- Printer Management
- Backup and Restore

As previously discussed, the primary SMFA's defined by ISO are evolving. The list presented above is the list of equivalent SMFA's from the X/Open literature. These classes represent an up-to-date mapping of major system management responsibilities.

Ongoing analysis is required to analyze these primary system management application classes and better understand the required functionality within each application class. This will be accomplished through both a top-down specialization of the ISO SMFA's and the DID216, and a bottom-up generalization of the proposed object and management services of OMG, OSF DME, ISO and X/Open.

5.3.3.5 OMG Object Services

The OMG object services recommended at this time is a composite of the object services required for both the DME and X/Open systems management architecture. These services include event notification, lifecycle, naming, persistence, synchronization, time, security, message catalog service (internationalization), relationships, concurrency, and transactions. This initial set will be mapped against availability from OMG and discussed along with the applications classes and management services in the next version of this paper. In parallel with the phasing analysis, continued analysis of all services will progress.

5.3.3.6 Management Service Classes

The recommended management services in the X/Open document included policy-driven base service, instance management service, policy management service, scheduling management service, and customization management service. These management services, the major ISO SMF's and the DME management services all previously discussed will be analyzed for potential inclusion to meet ECS systems management requirements. An example of a possible use of the X/Open collection service is provided below.

A Collection Service defines operations required to have two way reference relationships between objects. A collection allows logical groupings of objects to be queried or have operations applied. In a collection relationship, all collected objects know of all collections to which they are members, and all collections know of all objects collected in the collection. Each participant in the relationship maintains a set or list of the other participants in a relationship.

Because collections can refer to other collections, they can be organized into hierarchies. In addition, because collections are just groups of references an object can belong to any number of collections. The collection service is shown in Figure 47.

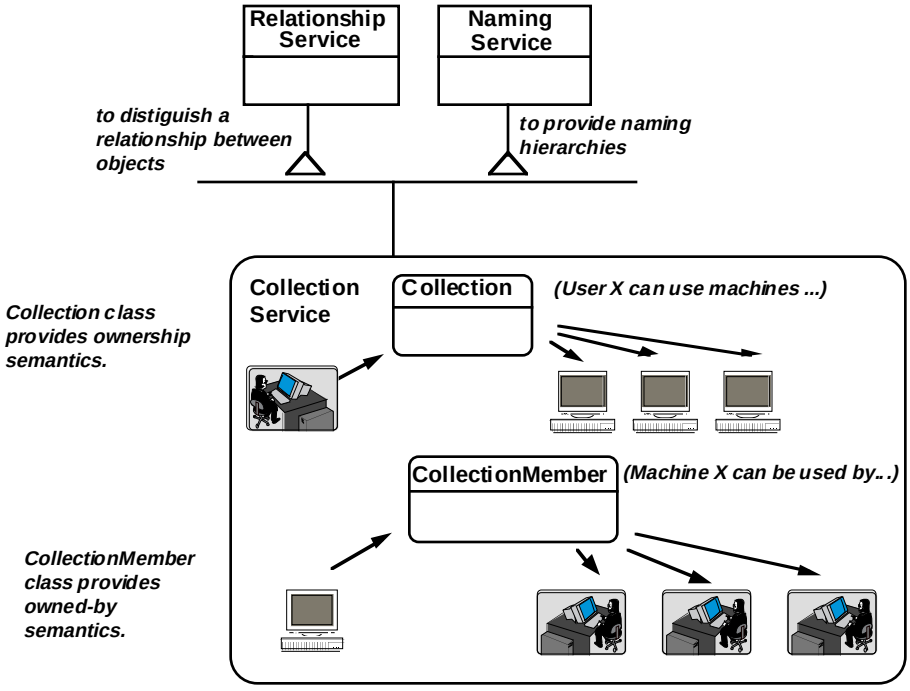


Figure 47. Collection Service

5.3.4 Managed System Object Classes

There is a growing consensus that the general GDMO model and format will provide a widely accepted and extensible model for managed objects. This type of model is useful in an integrated Framework architecture used for managing a large and diverse environment, such as ECS. This model is recommended for the systems management sub-architecture. Other organizations, including the ISO, CCITT, ANSI, IEEE, OSF, UI, X/Open, implementor's workshops (OIW) and even IETF, currently support the model. The GDMO model's support for true object-oriented mechanisms, including sub-classing (inheritance) and polymorphism have given management designers the flexible tools necessary to adequately manage of the array of resources in a large enterprise with thousands to thousand-millions of heterogeneous devices, hosts, operating systems, data objects and software objects. While this work is relatively recent, it is occurring rapidly in many areas. Management of many physical devices, which are integrated into another devices, such as disk drives, were previously believed to be unmanageable. Logical devices, such as file systems and processes, also presented a management challenge, with more complex relationships, states and transitions.

The primary managed system object classes will require additional analysis, but seem to center around the physical device class, the operating system class, and the application class. The physical device class defines the physical devices within the managed system. These are primarily based on the IETF HOSTMIB. This definition would need to be extended to cover the Display

Adapter, Disk, Network Adaptor, Printer, Processor, and Tape. The work done by DSIS is promising for these extensions.

The operating system class is fully defined for Unix by the GDMO Operating System Managed Object Specifications (OSMOS). It will inherit information from the HOSTMIB system table. The current definition covers most of the DSIS system attributes. It will support the UNIX Operating System as a subclass, inheriting characteristics from the higher-level Operating System class. The operating system class includes logical devices, processes, file systems, and software. Logical device work is ongoing to provide a mapping between logical devices, device drivers and mass storage. The mass storage work is in work by the IEEE Mass Storage working group. Processes defines running processes and is full defined in the GDMO Process Managed Object Specifications (PROCMOS). It also meets DSIS requirement, with certain exceptions. The file systems sub-class defines both local and remote file systems, and is fully defined by the GDMO File System Managed Object Specifications (FSMOS). It inherits information from the HOSTMIB. It also meets DSIS requirement, with certain exceptions. The software sub-class covers the concept of installed (i.e. dormant) software. It is primarily being worked on by POSIX 1003.7 and ISO. It should also inherit characteristics from HOSTMIB. This is also expected to cover the DSIS Registered File requirements.

The Application class of the managed system describes an application that can be run (even if it is not actually running). In that respect, it differs from a process, which only exists while the process actually exists. It has relationships to the underlying process(es) used to execute the application. It is expected that there will be a number of specializations made for specific applications such as backup/restore, or a print spooler application. A queue is a type of application. The managed object represents those aspects of a queue which are generic, but which would be inherited by specific queues, such as a spooler queue. This also covers the DSIS Print Spool queue requirements. The queue entry is a type of queue, and represents the generic aspects of an individual entry in a queue, such as a print request.

5.4 Issues

The major issues of large-scale distributed systems management have to do with performance, security, and policy. Performance, or lack thereof, will be a key challenge for the number of objects that require tracking in the ECS environment. Care will need to be taken with implementation to ensure choke points are designed into the system that would overload and impact system performance of the systems management coordination sites. The intelligent selection of node managers and cluster managers within an instance of a managed system can help alleviate this problem.

The system administrators will have an substantial amount of power at their consoles. Security precautions should be tight, and personnel must be trusted. The security design, due at PDR + 2 months, will need to fully examine the scope of distributed authority and understand the implications of safeguarding critical information dynamically distributed about the ECS.

The architecture as presented is policy-neutral. This will not be the case as the architecture transitions into design and implementation. This point in the ECS lifecycle is the critical point to make policy decisions that could have substantial impact to the cost and schedule baseline. These

decision points must be discerned and brought forward to ESDIS for resolution. Some of the high-level policy decisions that affect design include the placement of management authority, access considerations based on the generalization/specialization of user classes, configuration management practices, and the federation of administrative domains through contracts.

The placement of management authority examines who will have the authority to do what, and at what time (when). A possible approach to the problem might be to examine the major service classes and the major user classes, and make decisions of access to the services by the types of the user class. Systems management will be built on the concept of administrative domains. The domains are not necessarily the same for a given service type or user type. For instance, if a scientist should be able to traverse all DAACs of ECS without ever thinking about the resource costs, then an administrative domain can be set up for that user class (who may well be in a class of their own). Security, directory, and access control lists can be set up to accommodate the user class based on the policy decision made. A secondary policy decision based on this example would be this - if a user class has access to all sites, who actually has the authority to update the security, directory and access control lists at the sites the user is expected to visit? Each individual site, or GSFC? These are difficult issues, but decomposing the problem into class of services for classes of users may be a good place to start.

This same approach can help in the area of user account administration. Systems management will provide flexibility to aggregate users into logical groups, each group with unique implementations for user account structures, access capabilities, account credits, and password administration techniques. As heard at SRR, no scientist is going to want to fill out a registration form.

Appendix A. Open Distributed Processing (ODP) Discussion

A.1 Introduction

The fast growth of distributed processing in the industry has created an urgent need for a coordinating framework for the standardization of Open Distributed Processing. The concept of distribution is no longer limited to distributing information, but instead it has become a concept for the distribution of services and applications. In order to integrate service distribution, service interoperability, and service portability and to achieve the support for more advances in the capability, ODP logical framework architecture focuses on services as a collection of objects and building in numerous forms of transparency for technology insertion. ODP uses the existing standards (e.g. OSI standards) and introduces new constructs such as enterprise language, information language, computational language, engineering language, and technology language. In addition to the specification of ODP functions, the viewpoint languages, as illustrated in Figure A-1, can be used in new standards.

A.2 Reference Points

ODP defines four types of reference points, which exists at any interface of any object. They are:

- ***Perceptual reference point*** at which there is some interaction between the computerized part of the system and the outside world. This may be a human interface or a collection of robotic sensors and actuators. This reference point, for example, may be established in a graphics standard.
- ***Interworking reference point*** at which a communication interface can be established between systems, such as interconnection of physical media.
- ***Interchange reference point*** at which an interface to an external physical storage medium can be established.
- ***Programmatic reference point*** at which a programming interface can be established to allow access to a function. Example could be a notion of applications programming interface.

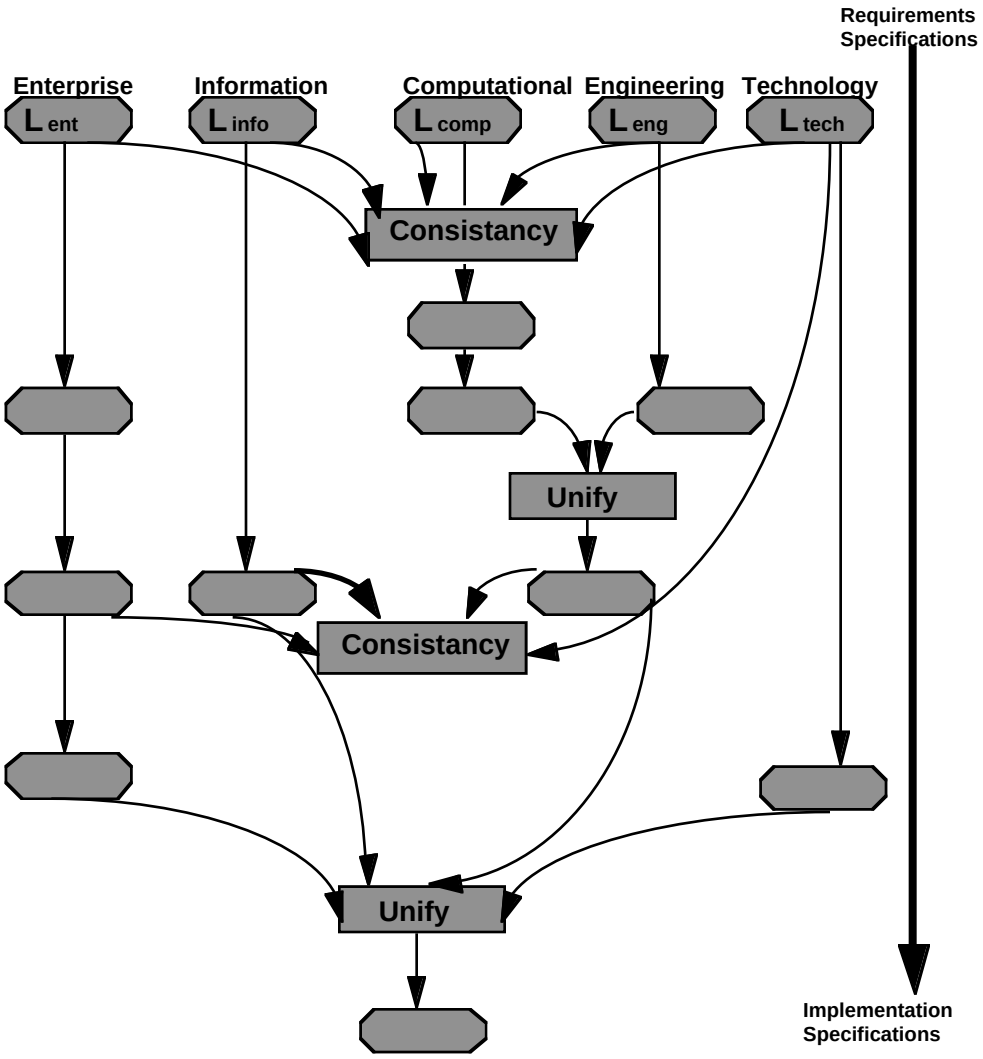


Figure A-1. ODP Viewpoint Languages

A.3 ODP Viewpoints

A viewpoint is an abstraction mechanism which allows a perception of a system which emphasizes a particular concern, while ignoring other characteristics that are temporarily irrelevant. The RM-ODP recognizes five viewpoints:

- ***Enterprise Viewpoint:*** concerned with the social, managerial, financial, and legal policy issues which constrain the human and machine roles that comprise a distributed system and its environment

- ***Information Viewpoint:*** concentrates on information modelling, flow and structure, and information manipulation constraints
- ***Computational Viewpoint:*** focuses on the structure of application components and the exchange of data and control among them
- ***Engineering Viewpoint:*** concerns the mechanism that provide distribution transparencies to application components
- ***Technology Viewpoint:*** focuses on constraints imposed by technology, and the realized components from which the distributed system is constructed

The ODP work recognizes that viewpoints do not comprise a layered architecture. There is no inherent ordering among the viewpoints and there is no implied methodological sequencing. Viewpoints are qualitatively different from on another. The computational, information, and enterprise viewpoints are often used to derive the programming interface.

A.4 Benefits of ODP Standards

Using these ODP standards, the enterprise, the system implementor, the end-user etc., can benefit with the:

- provision of availability, reliability and fault tolerance;
- reconciliation of constraints on the location of users with constraints on the location of computing resources;
- integration of systems with different resources and different performance;
- improvement in performance as a result of parallel operation of different applications or parts of a given application;
- provision of modularity that allows incremental growth without impacting existing applications;
- provisions for sharing, integration or partitioning of resources and applications across different systems, management domains and locations in response to user needs;
- provision for the containment of cost and risk with new modules for load balancing and dynamic reconfiguration;
- distribution of system management by allowing decentralized management environment;
- provision of security including authentication and access control facilities;
- maintenance of data integrity and consistency, auditing, software development engineering tools to support the development of distributed applications.

The most important benefit for using ODP standards is a provision for masking the technical details of distribution from application programs covering:

- ***access transparency*** - masks differences in data representation and invocation mechanisms between heterogeneous computer architectures

- ***location transparency*** - masks changes in configuration of application components, and enables the transfer of configuration-independent interface references between components,
- ***replication transparency*** - hiding the effect of multiple copies of services and information,
- ***concurrency transparency*** - masks the scheduling of invocations of operations that act on shared state
- ***federation transparency*** - masks interworking boundaries between separate administration domains and heterogeneous technology domains
- ***liveness transparency*** - masks the automated transfer of components between active and passive states
- ***migration transparency*** - masks the dynamic relocations of components from both the components themselves and their clients
- ***failure transparency*** - masks recovery of failed components,thereby enhancing fault tolerance
- ***resource transparency*** - masks variations in the ability of the local ODP infrastructure to provide the resources for an application component to engage in interactions with other remote components

A.5 Engineering Model

The engineering model of ODP is especially relevant to the communication and systems management architecture. Interaction between two basic engineering objects in different address space requires a configuration of engineering objects called a channel. A channel is the medium through which remote interactions pass. Figure A-2 is a simplified view of a channel, other objects may be involved depending on the level of transparency required.

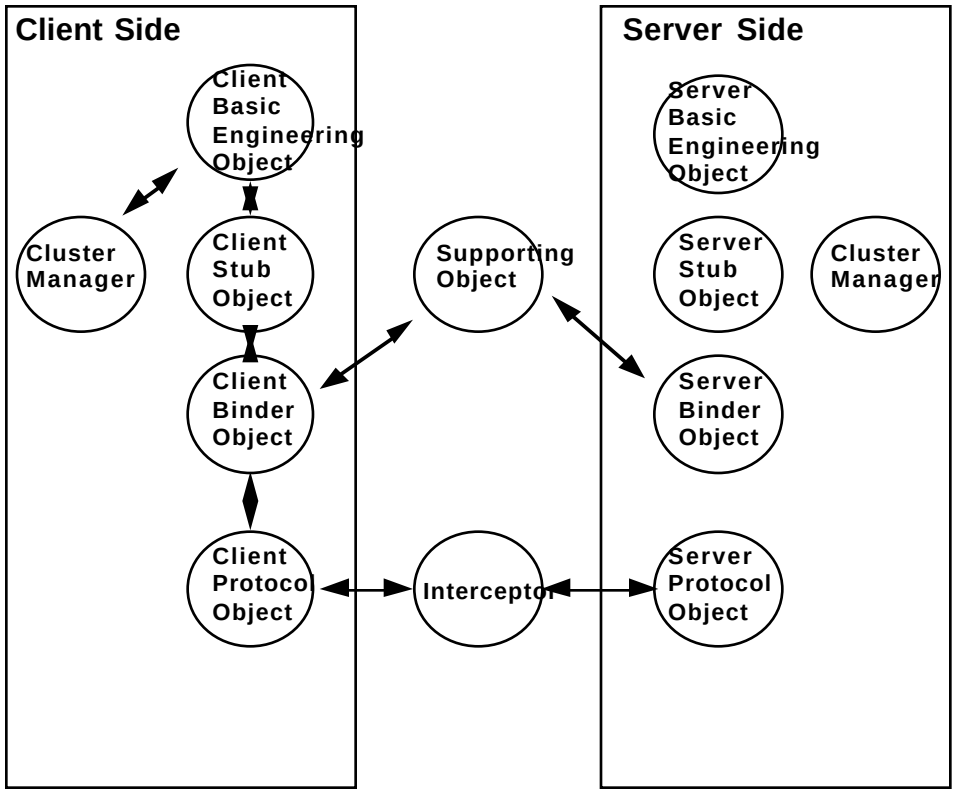


Figure A-2. Simplified Engineering Model

Distribution transparency is provided through the contribution of stub and binder objects. Binder objects control the binding of server interface for a client interface. Subtypes of binders provide location transparency, replication transparency, failure transparency, migration transparency and resource transparency. Stub objects differ from binder objects in that the actually modify the information exchanged across the channel. Subtypes of stub objects provide concurrency and access transparency.

The protocol objects hide the actual communication mechanisms between systems. The interceptor objects lie at administrative or technology boundaries and provide protocol and type conversions.

A node in RM-ODP is a group of engineering objects that share common processing, storage and communication resources. A node can contain one or more capsules. Capsules correspond to address spaces and are created by a factory object and managed by a nucleus object. RM-ODP capsules are the OMG equivalent of an Object Adapter. An Object Adapter viewed this way would be precluded from running across more than one node. The nucleus object is responsible for establishing channels and for the creation of threads. The nucleus object must be statically connected to a trader which may actually reside at another node. The ODP nucleus is an OMG equivalent of an ORB.

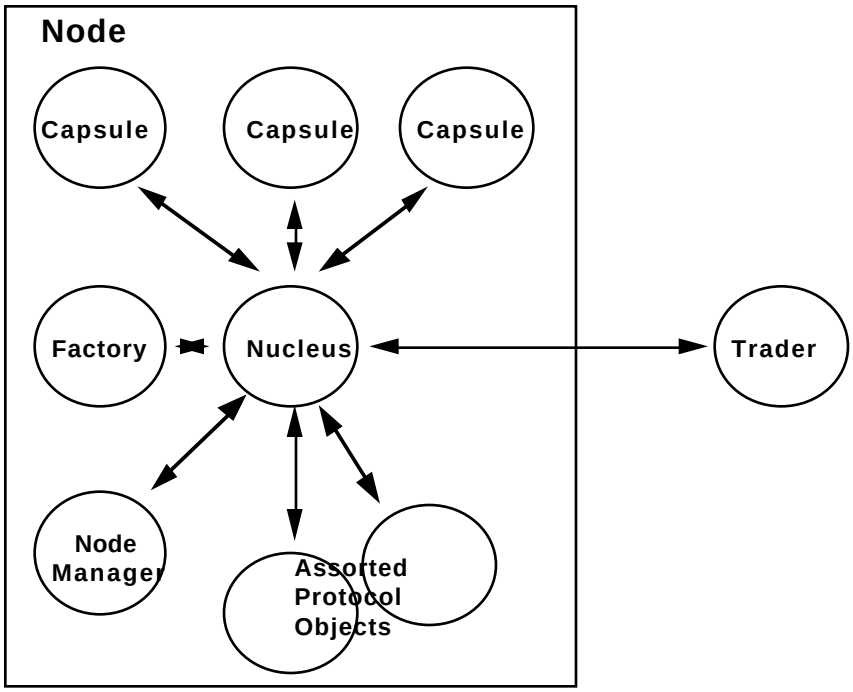


Figure A-3. ODP Node

Cluster objects are basic engineering objects grouped as units of activation, deactivation and migration and correspond to a segment in virtual memory systems. The interfaces for these objects must be bound to each other or to channels. These clusters may be instantiated through templates into a capsule. A cluster manager is required for every cluster and every capsule requires a capsule manager. The capsule and cluster managers provide instantiation, activation, deactivation, reactivation and checkpointing of the clusters.

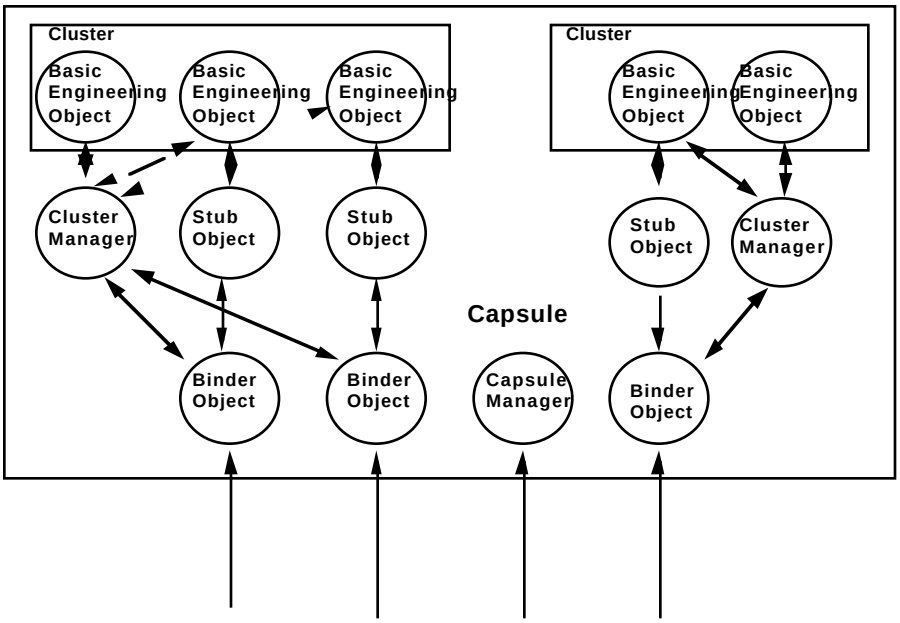


Figure A-4. ODP Capsules and Clusters

Appendix B. OSF Distributed Computing Environment

B.1 Introduction

Distributed Computing Environment (DCE) is an enormous software system from the Open Software Foundation (OSF), embodying some novel and complex concepts. It provides most reliable and cost-effective services and tools that support the creation, use and maintenance of distributed applications in a *heterogeneous* computing environment. Location transparency, Database and user interface consistency, effectiveness in terms of speed of response and extensibility, reliability, fault tolerance and recoverability, security, availability, data and resource sharing, interoperability are some of the key benefits of using OSF distributed computing environment (DCE).

DCE is based on three distributed computing models: client/server, remote procedure call, and data sharing. The following sections briefly describe each model.

B.1.1 The Client/Server Model

Distributed systems rely entirely on computer networks for the communication of data and control information between the computers of which they are composed. Client/Server computing results when you establish a set of procedures or subroutines as an independent program (server) on one machine and make it possible for client programs on possibly other machines to invoke the procedures, much as if they were local to the client program. As illustrated in the following figure, typically the client side of the application resides on the node that initiates the distributed request and receives the benefit of the service. The server side of the application resides on the node that receives and executes the distributed request.

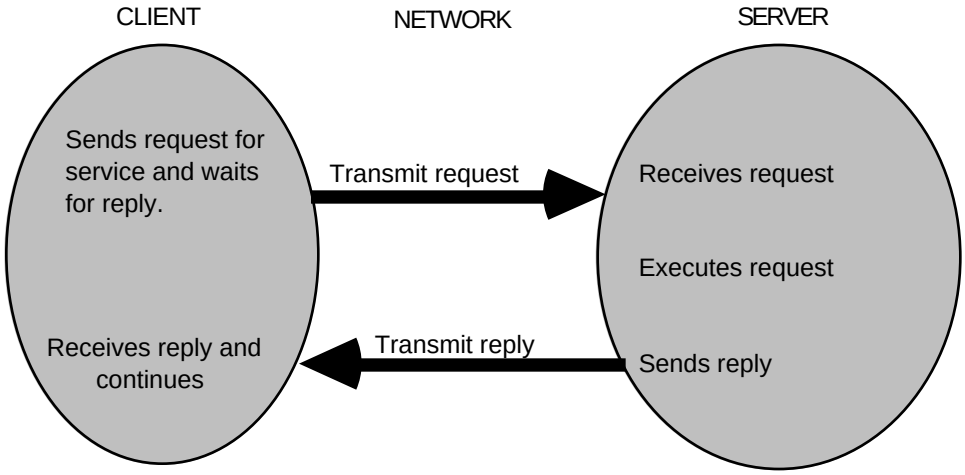


Figure B-1. Client/Server Model and Sequence of Communication

It is important to note that the terms *client* and *server* should be viewed as relative roles rather than as absolutes. In other words, server for one request from a client, itself can be a client to another server for different request and vice versa. For instance, client communicates to print server to print a file. This print server could be a client to a file server to get the file for printing. Typically, many nodes need to be able to run the client side of an application, whereas only one or two nodes may be equipped to run the server side of an application. The DCE services are themselves examples of distributed programs with a client and a server. The basic communications mechanism described next assumes the presence of a client and a server.

B.1.2 The Remote Procedure Call (RPC) Model

In this model, the client calls a procedure on a remote machine as if it were a local procedure call. RPC provides a high-level programming model to the distributed application programmer, hiding communications details and removing non-portable system and hardware dependencies. RPC also converts data to and from forms required by different kinds of hosts and it manages automatic recovery from network or server failure. DCE RPC, is an implementation of this model. Most of the other DCE components use it for their network communications.Following Figures illustrates this model.

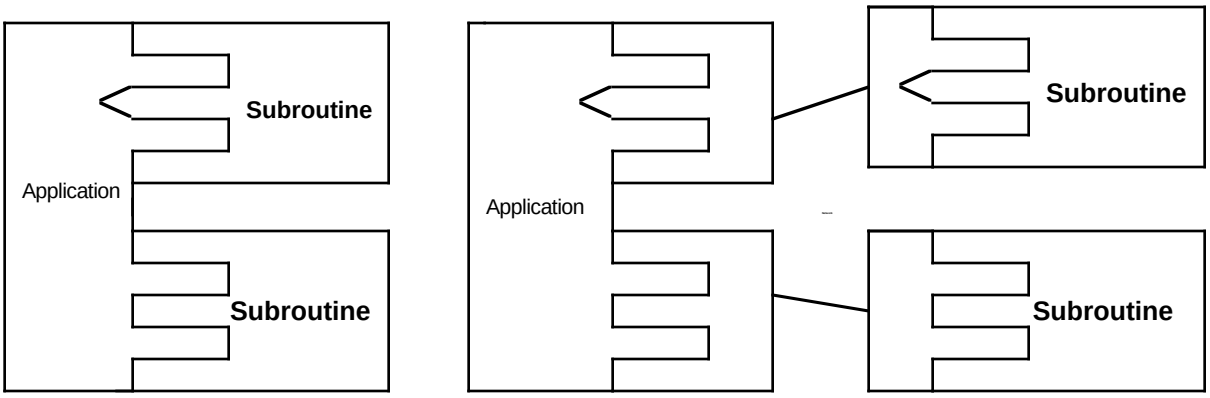


Figure B-2. Remote Procedure Call

DCE RPC facility consists of both a development tool and a runtime service. The development tool consist of a language(and its compiler) that supports the development of distributed application as illustrated in the following figure. The runtime service implements the network protocols by which the client and and server sides of an application communicate. DCE RPC also includes software for generating unique identifiers, which are useful in identifying interfaces and other resources.

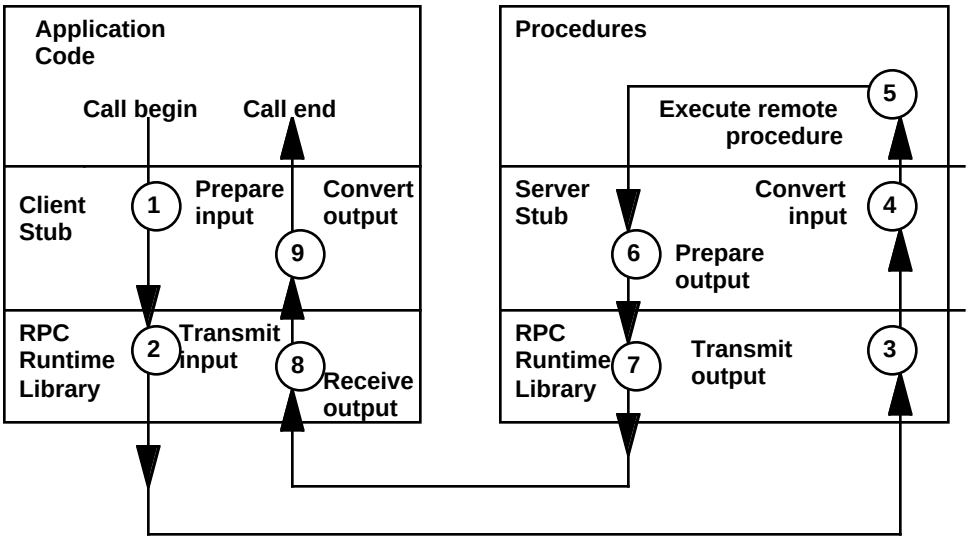


Figure 3.2.4.2 (B) Role of RPC Interface

Figure B-3. Role of RPC Interface

The DCE RPC is made up of interface definition language (idl), uuidgen utility, runtime library routines, error-handling routines, rpccp and rpc deamon.

B.1.3 The Data Sharing Model

Some of the DCE services are based on the data sharing model, in which data is shared by distributing it throughout the system. Data sharing focuses on distributed data rather than distributed execution. Data sharing usually entails having multiple copies of the same data. DCE includes mechanisms for keeping copies of data consistent. Furthermore, it takes care of synchronizing multiple access to data. Two DCE services are based on the data sharing model. Directory Services (CDS and GDS), maintain caches on the client. The caches contain copies of data that users on the client have recently accessed. Subsequent access to the data can be made locally to the cache, rather than over the network to the server. Distributed File Service (DFS) is also based on the data sharing model. A DFS client maintains a cache of files that have recently been accessed by a user on the system. DFS server distribute and revoke tokens, which represent a client's capability to perform operations on files. Through careful token management, the DFS server can ensure that its clients do not perform conflicting operations or shared files, and that they do not see inconsistent copies of the same file. With this model, a user anywhere in the distributed system can share data by placing it in the namespace or in a file, whichever is appropriate for the application. Data sharing like RPC, enables users and programmers to communicate transparently in a distributed system. Figures B-4 and B-5 illustrate the data sharing model.

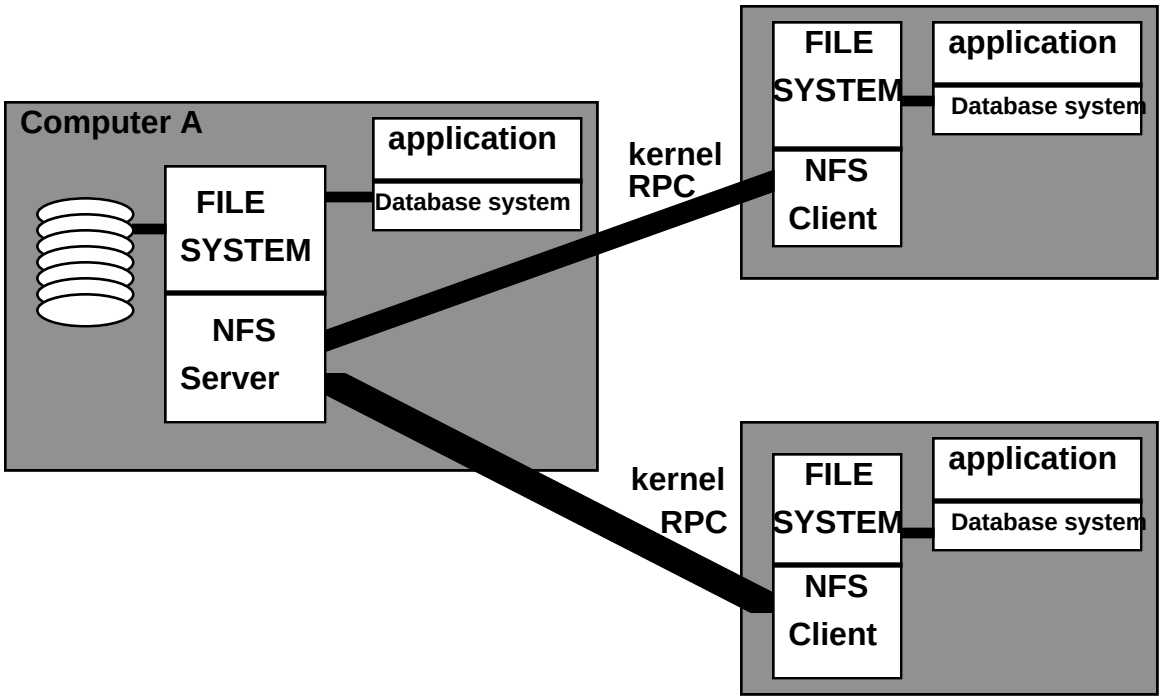


Figure B-4. Data Sharing Model in a closed system.

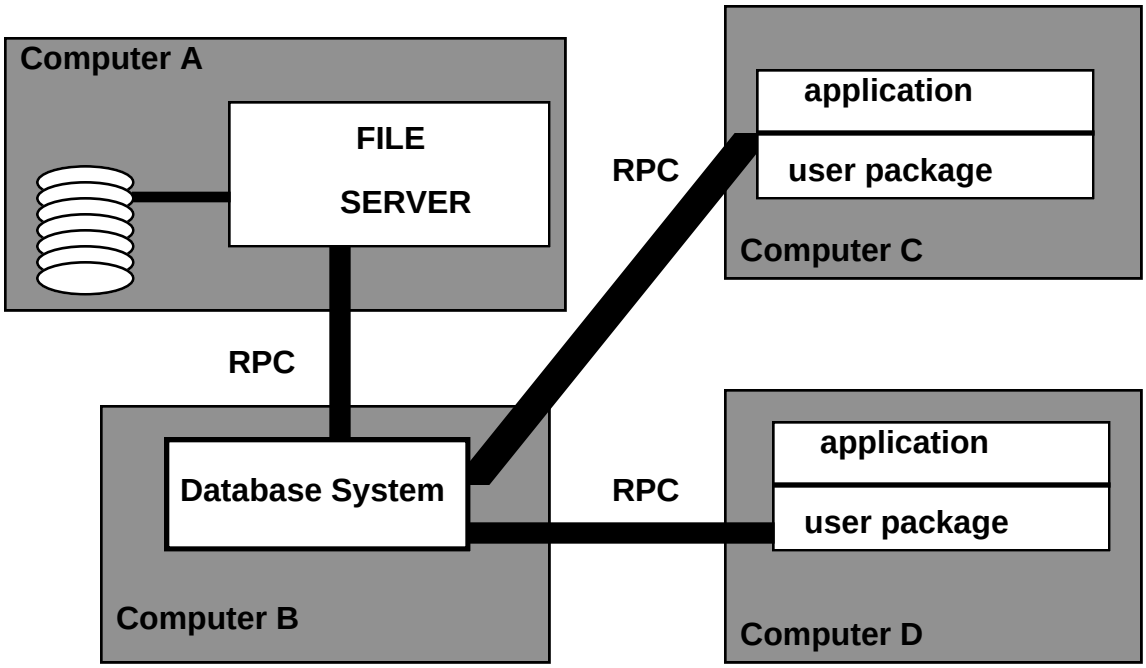


Figure B-5. Data Sharing Model in a distributed system.

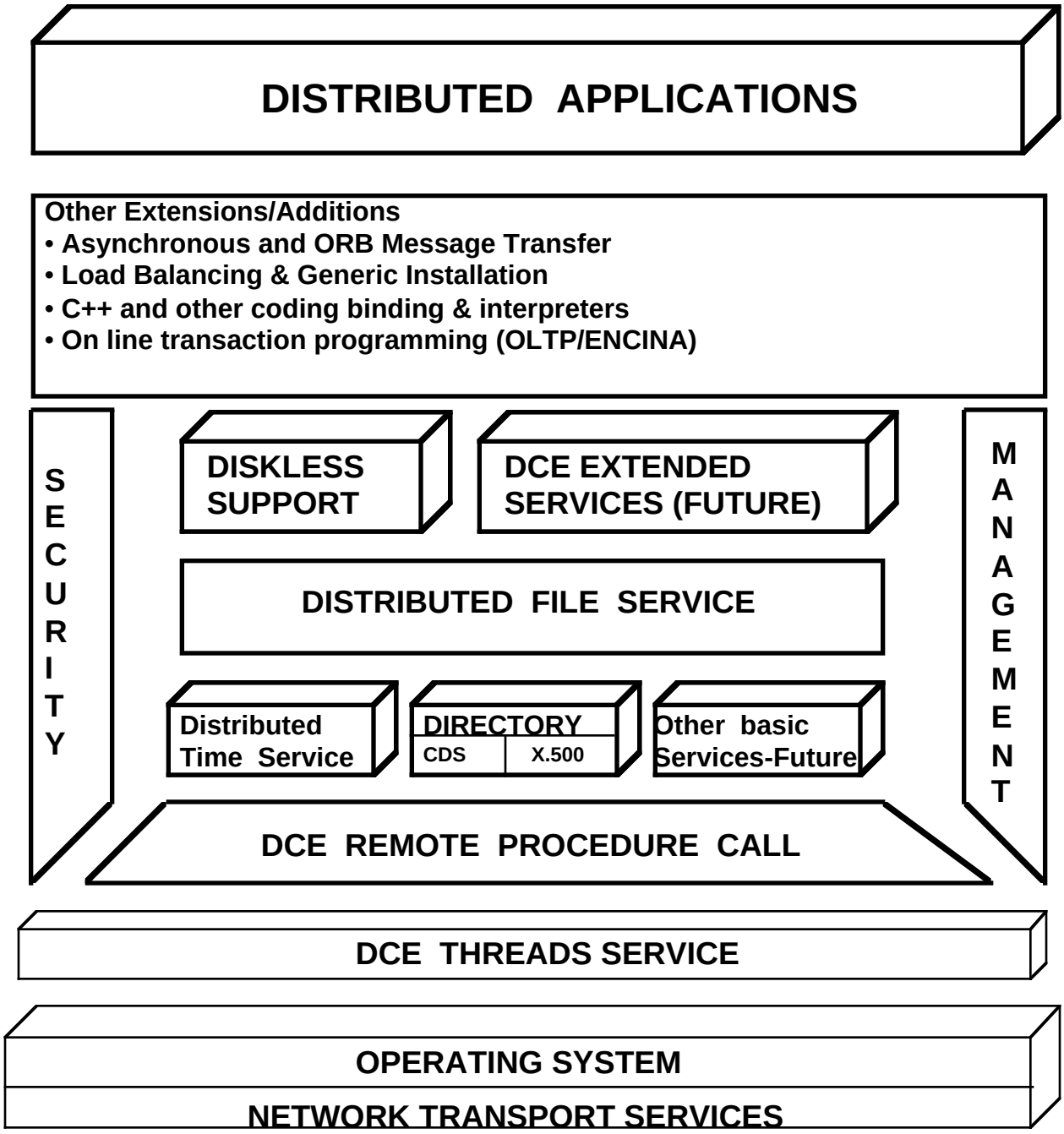


Figure B-6. DCE Architecture

B.2 Architectural Overview of DCE

OSF's DCE is a layer between the operating system and network, on one hand, and the distributed application on the other. DCE, as a layer of software, provides the services that allow a distributed application to interact with a collection of heterogeneous computers, operating systems and networks as if they were a single system. Figure B-6, one conceptual model of DCE, shows the relationship of the DCE distributed services (Security, Directory, and time) to RPC and Threads services.

DCE Threads supports the creation, management, and synchronization of multiple threads control within a single process. Threads on a client can control separate remote procedure calls to various server hosts on the network. Threads on a server enable the server to handle multiple client requests concurrently.

The idea behind **DCE Remote Procedure Call(RPC)** is simple: to allow a program to call procedures in a different computer or in a different address space in the same computer. The rpc call is sent in the form of a *request message* to a remote process that is able to receive the call, execute the procedure and send back a *reply message.*

The **DCE Directory Service** is the central repository for information about resources in the distributed system. Typical resources are users, machines, and RPC-based services. Both CDS(Cell Directory Service) and GDS(Global Directory Service) are accessed using a single directory service application programming interface API, the X/OPEN Directory Service(XDS).

The **Distributed Time Service** provides synchronized time on the computers participating in a Distributed Computing Environment. DTS synchronizes a DCE host's time with Coordinated Universal Time(UTC).

The **DCE Security Service** provides secure communications and controlled access to resources in the distributed system. These services are comprised of Registry service, authentication service, the privilege service, the access control list facility and login facility.

The **DCE Distributed File Service(DFS)** allows users to access and share files stored on a File Server anywhere on the network, without having to know the physical location of the file.The DFS achieves high performance, particularly through caching of file system data.

The **DCE Diskless Support Service** provides the tools that allow a diskless node to acquire an operating system over the network, obtain configuration information, connect to DFS to obtain the diskless node's root file system, and perform remote swapping.

The **DCE Management** shown in the figure B-6 is not a single component, but a cross section of the other components. Each DCE service contains an administrative component so it can be managed over the network.

B.3 Motivation for using DCE and industry support justification

DCE provides interoperability and portability across heterogeneous platforms. In Europe and Asia many companies are utilizing DCE's DFS and GDA features very widely. Any computing organization comprising several cooperating hosts can benefit greatly from the administrative

support afforded by a DCE environment. For example, in DCE the database of computer users and their associated information (such as passwords) can be administered centrally, removing the need for an administrator to update information on every single node in the network each time a new user is added. Beyond HP, IBM, and DEC, many companies such as Attrium, Tivoli, Hal PC, Transarc, Ellery, and OEC, are providing tools and libraries to DCE, furthering the rationale for selection of DCE as a foundation for advanced distributed processing.

Appendix C. SNMP Management Notes

C.1 SNMP MIB

uses a tree to store managed information

is static database, MIT is determined at design time. Potential difficulties in handling composite device structures, where different components may require their own database models that cannot be unified into a single MIB due to its static structure

No means to directly define relationships between MIB entries

SNMP v.2 resolves some of the above limitations. It can collect sets of data and it places intelligence "in the network"

C.2 RMON

IETF developed, has three parts

* SMI, describes and names objects to be managed

* MIB, defines attributes and info regarding SMI objects

    - MIB1 and MIB2 are internet standards

    - Private MIBs may exist

    - RMON MIB provides distributed monitoring and analysis

    * SNMP, communications mechanisms between agent and manager

    - RMON MIB collects network info. from remote devices (segments)

    - Nine attribute classes:

        * Statistics (fragments/collisions)

        * History (stats over time)

        * Alarms (compare current stats to thresholds)

            * Hosts (information on active hosts)

            * HostTopN (info on highest host in particular stat)

            * Matrix (traffic between nodes)

            * Filter (packet filtering by equation)

            * Packet Capture (provides packet capture)

            * Event (controls events form a device)

C.3 RMON-MIB

improvement to SNMP MIB, provides improvement in intelligent monitoring

developed by the Internet Engineering Task Force

requires less network traffic and overhead than SNMP proxy agents

primarily for network rather than system management tasks

Appendix D. Notes on ISO/CCITT

D.1 OSI Systems Management Framework

recognizes three management approaches

* Protocol Mgmt. - management within a communications protocol

  - protocol internal mechanisms to control a particular instance of communication

  - example is transport flow-control windowing

* Layer Mgmt. - management within a layer (with several protocol instances)

  - layer specific networking services to manage layer specific resources

  - example is network layer routing mechanisms, link layer token control

* Systems Mgmt. - management across layers

  - management of resources associated with multiple layers and instances

D.2 OSI Management Communication Model

- connection oriented transport required

- agents and managers are viewed as peer applications using services of CMISE

  using symmetric organization

- CMISE provides SAPs to support controlled associations between agents and managers

  for get/set/action and event-notification operations:

  * Management Communication Services

    M-Initialize: Establish management association

    M-Terminate: Terminate management association

    M-Abort: Unconfirmed termination

  * Management Information Tree Operations

    M-Create: Creates an MO instance record in the MIT

    M-Delete: Deletes an MO instance from MIT

  * Managed information manipulation services

    M-Get: Retrieve information

    • Aggregate (bulk) and selective (filtered) retrieval

    M-Cancel-Get: Cancel retrievals

M-Set: Change an attribute value

M-Action: Invoke an MO operation

M-Event-Report: Generate an MO event report to a manager

- CMISE uses ACSE and ROSE to support services

- Agents perform:

  * Get/Set/Action selection

  * Event detection

  * Event forwarding and discrimination processing

    - forward is to managing entities enrolled to get events, from MIT records

- Managed Object (MO):

  * provides notifications

  * performs operations

  * contains attributes

  * is a superset of OO data model, which has only operations and attributes

- Note bulk and selective retrieve are not provided by SNMP, but SNMP2 will offer bulk.

- Note action command for explicit operation invocation is not available in SNMP (must use Set command with a shell script built from the set to invoke an operation - the limited set command provides less capability in passing parameters and in synchronizing invokes)

D.3 CMIP MIB

- Agent maintains Management Information Tree (MIT) database

- MIT has numerous managed objects (MOs)

- CMIP can create, delete, retrieve, or change MOs in the MIT, invoke operations, or receive event notifications

- MOs are hierarchically arranged in MIT, similar to X.500 directory tree

  * Relative distinguishing name (RDN) defines instance of MO

    and is part of the instance attributes

  * RDN concatenated with MIT path from root to node is unique DN

  * MOs provide notifications, perform operations, and contain attributes

- MIT is dynamic database

  * MIT is determined dynamically

  * Provides flexibility and efficiency in managed information access

* Managing entities can control the contents and structure of the database

* Significant implementation difficulties

- resources to store and process cannot be predicted at design time

* Changes in MIT may result in corruption of the database

- MO could be deleted while other MOs have pointers to it

- each application needs to build and maintain its own subset

- explicit means to represent relationships between MOs is available

* provides efficient means to correlate related data items for analysis

- broadcast storms

- IP down failures in a string

D.4 Fundamental Steps to Build an OSI Managed Agent

1. Identify class structure and inheritance relations among managed objects

- identify similarities of managed elements

- capture similar elements into MO classes

- develop MO hierarchy and inheritance relationships

2. Design and specify MO syntactical structures using GDMO

- look first for standardized MOs from standards committees

- define managed attributes, operations, and event notifications for each MO class

3. Design generic MIT structure for the device

Appendix E. Notes on GNMP

E.1 GNMP by NIST

- is an integral part of OMNI*Point* 1
- uses OSI standards complaint protocols and services (CMIP/CMISE)
- has management information definitions
    * layer 1 and 2 network functions of the OSI basic reference model
    * layer 3 and 4 also included in part
    * includes the OIW MIL, aligned with NMF
    * future layers, new stable standards definitions with future releases
        - layer 3 through 7 in Version 2
        - definition of applications and services outside of OSIRM
            - computer operating systems
            - database management systems
- has seven systems management functions
    * Object Management Function
    * State Management Function
    * Attributes for Representing Relationships
    * Alarm Reporting Function
    * Event Report Management Function
    * Log Control Function
    * Security Alarm Reporting Function
- has two optional peer entity authentication modes

    1 username and password fields of simple credentials using ACSE service and protocol for definition of a new functional unit (authentication). Authentication is compared against authorized users list - passwords are transmitted in the clear. Username and password distribution is beyond scope of GNMP

    2 all of mode 1, with a hash function applied to the authentication information. Time stamp may be included. Hash function is the Secure Hash Algorithm, but other hash algorithms may be supported (MD5 in part 12 of stable

implementor's agreements). As with mode 1, username and password distribution is beyond scope of GNMP

- Security Features
    * Authentication - peer entity and data origin authentication
    * Access Control - currently full access to authenticated associations
    * future features
        - authentication - peer entity and data origin authentication
        - access control - access control mechanisms within authenticated association
        - confidentiality - connectionless
        - integrity - connectionless
- primary source of specifications is part 18 of the OIW Stable Implementation Agreements
    * provides implementation specifications for network mgmt. based on CMIP/CMISE
- additional specifications include:
    * IEEE 802.1B LAN/MAN Management
    * IEEE 802.3 Repeater Management
    * ANSI X3T9.5 FDDI Station Management
    * CCITT Study Group IV Generic Network Information Model
    * ISO/IEC JTC1/SC6 Transport and Network Layer Management
    * Network Management Forum
- GNMP acts as a manager's manager: building a hierarchical NMS with std. exchange between integrators, and between integrators and managers.
- GNMP does not include analysis of management information and HMI requirements
    * only addresses interoperability between network management components
- GNMP method to build interoperable NMSs:
    1. Develop plan for partitioning network management responsibilities
        - number, location, size, and scope
    2. Authentication determinations (where, what)
    3. GOSIP application requirements for each manager and integrator
    4. Managed object selections for each manager and integrator
        - determination of need optional attributes and conditional packages
        - name binding determination for relationships between objects

5. Specification of protocol requirements for each integrator and manager

- NIST GNMP Users Guide under development

E.2 GNMP Conformance Requirements

* Management Communications

1. satisfy part 18, clause 8.3.1 of June 1992 Stable Implementor's Agreements

2. provide ACSE services and protocols as in GOSIP v.2, section 4.2.7.1 and modified per part 18, clause 6.5 of Stable Implementor's Agreements

3. provide ROSE services and protocol as in ISO 9072-1 and ISO 9072-2, and modified per part 18, clause 6.5 of Stable Implementor's Agreements

4. support presentation and session layer services per part 5, clause 13.7 of the June 1992 Stable Implementor's Agreements.

5. VT, FTAM and MHS as required to be in compliance with GOSIP v.2 (sections 4.2.7.2, 4.2.7.3, 4.2.7.4, 5.3.1 and 5.3.2).

* Management Information

1. include at least one managed object

2. selected, where applicable, from MO definitions in referenced documentation and implementation agreements:

- Definition of Management Information
- IEEE 802.1B LAN/MAN Management
- IEEE 802.3 Repeater Management
- ANSI X3T9.5 FDDI Station Management
- CCITT Generic Network Information Model
- ISO/IEC JTC1/SC6 Management Information related to OSI Network Layer Standards
- ISO/IEC JTC1/SC6 Management Information related to IS to IS Intra-Domain Routing Information Exchange Protocol
- ISO/IEC JTC1/SC6 Management Information related to OSI Transport Layer Standards
- Annexes A and B of part 18 of OIW SIAs
- NMF Management Information Library

- ISO/IEC 10164-x System Management Functions

3. MO specification shall include:

a. document from which MO is selected

b. optional attributes and conditional packages required

c. at least one name binding for each MO selected

4. MOs not from referenced documentation shall satisfy part 18, clause 8.3.3 of the stable implementor's agreements

5. MOs not from referenced documentation shall use techniques and templates specified in GDMO

6. MOs and elements of MOs not from referenced documentation shall be built as possible using MOs and element of MOs from the referenced documentation as superclasses

7. All new MOs shall have registered object identifiers and must be publicly available

8. Management Information Catalog may be used for identification of additional managed object definitions

* Systems Management

1. satisfy the requirements for systems management in part 18, clause 8.3.2 of the OIW stable implementor's agreements

2. selected systems management functions from part 18, clause 8.3.2 shall be identified

3. Agent role, manager role, or both roles shall be specified for each system management functional unit selected

* Security

1. Mode 1 or Mode 2 peer-entity authentication shall be specified

2. ACSE extensions shall be used during association establishment for peer-entity authentication per ISO 8649 and 8650

3. Authentication shall use simple credentials per part 3 of ISO 9594 for use in the authentication field of the ACSE PDU.

4. Simple authentication as defined in part 8, section 2 of ISO 9594 shall be used

5. Directory is not mandatory, therefore, authentication functionality shall be performed by the authenticating entity.

6. Password usage shall conform to FIPS Pub. 112

E.3 GNMP Testing

* Conformance Testing
    - Products will be tested in accredited GNMP testing laboratory
    - compliant products will have registration in the conformance tested GOSIP product register
* Interoperability Testing
    - commercially available interoperability testing services
    - on site multi-vendor testing
    - interoperability test and registration service register of the GOSIP register database will maintain list of recognized interoperability services
    - minimum interoperability test suites will be identified and maintained by the U.S. GOSIP Testing Program

E.4 GNMP and SNMP

* Future work item to integrate GNMP and SNMP into a single NMS

E.5 GNMP Ensembles

* MOs and SMFs that solve a particular management problem
* Provides
    - Problem to be solved
    - Requirements associated with the problem
    - Solution to the problem
    - Standards and MOs making up the solution to the problem
* NMF developed concept and is developing specific ensembles:
    - OSI Interworking ensemble
    - Reconfigurable Circuit Service: Configuration Management (RCS) ensemble
* NIST will consider future inclusion of ensembles to GNMP

E.6 Documentation

v.1 GNMP specification

anonymous ftp (osi.ncsl.nist.gov or 129.6.48.100); type ftamosi

./pub/gnmp/gnmp.asc	--ascii
./pub/gnmp/gnmp.ps	--Postscript
./pub/gnmp/gnmp.W51	--WordPerfect 5.1

Postscript figures are in same subdirectory as figN.ps where n=1,2,3

NIST stable implementors agreements

anonymous ftp (osi.ncsl.nist.gov or 129.6.48.100); type ftamosi

./pub/oiw/agreements/XS-9112.asc	--ascii
./pub/oiw/agreements/Xs-9112.w51	--WordPerfect 5.1

Additional documentation

NTIS	1-703-487-4650
IEEE Computer Society Press	1-800-272-6657
Network Mangement Forum	1-908-766-1544
ANSI	1-212-642-4900
ISO/IEC	41-22-749-01-11

Appendix F. Notes on OSF DME

Note: DME has been dropped at OSF. The material below is for reference only.

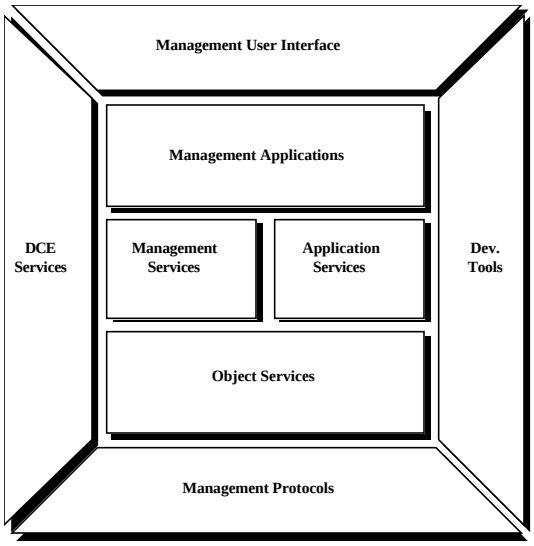


Figure F-1. DME Architecture

The OSF DME supports the ECS System Management sub-architecture objectives by providing the foundation of a productive and cost-effective enterprise management of heterogeneous distributed environments by providing the following features:

- Consistent user interface
- Stable and uniform services
- Consistent architectural framework and APIs for integration between applications
- Platform, network, operating system and vendor independence
- Flexible and scalable to local management policies such as centralized or distributed
- Interoperable, supporting multi-vendor networks to appear as a single logical entity, enabling the transparent implementation of software applications on the entire configuration.

The DME Architecture, including DME Services and Third Party/User-developed applications will utilize the OSF Motif Graphic User Interface in its Management User Interface (MUI), providing consistency across all applications.

The OSF DME architecture, consists of three major components:

- Management Services provided by the (Network Management Option or NMO)
- Object Services provided by the (Management Framework or MF)
- DME Distributed Services or DS, consisting of:
  - License Management Service (LMS)
  - Software Distribution Service (SDS)
  - Event Services (EVS)
  - Subsystem Management Service (SMS)
  - Personal Computer Services

The DME architecture provides the following Distributed Service (DS): License Management Service (LMS), Software Distribution Service (SDS), Event Services (EVS), Subsystem Management Service (SMS), and Personal Computer Services. These services have been released as DME version 1.0.

F.1 DME Distributed Services

F.1.1 OSF DME License Management Service (LMS)

The DME License Management Services (LMS) as depicted in Figure 4-1, utilizes the services of OSF DCE to manage and control access to software licenses in a distributed environment. The DME License Management Service consists of the License Server, the License creation tool, the Client library, and the License Manager. The DCE Cell Directory Service (CDS) allows licensed applications and license management servers to locate one another. The DCE RPC facilitates communication between licensed applications and license servers. In addition, the DCE Security Service is used for access control. The OSF DME License Management Service provides a vendor-neutral mechanism for creating, distributing and using software licenses, including the following features:

- Security
- Flexibility of Usage
- Termination Control
- Designation of Licensed Users
- Central Administration
- Security Through a Time Stamp
- Metering of License Usage
- License Queuing
- Access from a PC

The base DME technology for this services was submitted by Hewlett-Packard.

F.1.2 OSF DME Software Distribution Service (SDS)

The OSF selected Hewlett-Packard's Software Distribution Utilities for the basis of the DME Software Distribution Service. OSF adapted the HP SDU package to the DCE Remote Procedure Call. The Distributed Management Environment SDS automatically manages the packaging, distribution, installation, and management of software, and supports variety of media, including CD-ROM and tape. SDS supports a single methodology, irrespective of the underlying OS for deploying, updating and controlling software in a heterogeneous environment. SDS uses OSF DCE services, including RPC, Security, and Cell Directory Services.

The DME Software Distribution Service also supports highly flexible administrative policies regarding software maintenance.

The technology utilized in the OSF DME Software Distribution Service has been adopted as the basis of POSIX 1003.7.2 and conforms to draft 8 of that standard.

F.1.3 OSF DME Event Services (EVS)

The OSF DME Event Services delivers a basic mechanism for consistently handling events in a distributed computing environment. It provides a common way for both system and user applications to generate notifications of events and forward them to a destination anywhere in the distributed environment. Applications generating events need not worry about how, when, or where those events will be delivered and consumed.

The OSF DME EVS provides a consistent notion of time, security and context by utilizing DCE services. A single logging mechanism for all components in a distributed environment is supported by DME Event Services. Event Services provides notification of problems and changes occurring in system, including forwarding, logging and filtering. Powerful, programmable filters analyze attributes of event notifications, and include the capability to associate events with actions. The DME Event Services also provide a high-level template language for event definition. EVS utilizes DCE Threads, RPC, Cell Directory Service, Distributed Time Service, and Security Service.

The OSF continues to track standards of the ISO and other relevant standards regarding event reporting and management, and plans to incorporate relevant standards as they emerge. The base DME technology was submitted by Banyan Systems, Inc., who acquired the technology from Wang.

F.1.4 OSF DME Subsystem Management Service (SMS)

The OSF DME Subsystem Management Service provides a consistent way to inquire about the status of subsystems and to shut them down in an orderly fashion. SMS has a dependency mechanism to control the order in which subsystems are started and stopped. This capability can be used to ensure that a subsystem will not start until subsystems on which it depends are started. The capability to notify an administrator of abnormal termination of subsystems is also provided.

F.1.5 OSF DME Personal Computer Services (PCS)

The OSF DME Personal Computer Services Provides efficient full integration and management of DOS-based PCs from a POSIX-compliant system. The Personal Computer Services supports PC Ally which provides a communications path between a PC and workstation that is suited to the limitations of the MS-DOS environment. The PC-LMS (PC-License Management Service), PC-PRS (Print Services) and PC-Agent utilize PCS Ally as a reliable communication mechanism which allows PC applications to communicate with systems running DME. The DME Personal Computer Services supports network licensing, event notification services, fault monitoring and PC configuration management services. The Personal Computer Services are not currently an ECS requirement.

F.1.6 Print Services

MIT's Palladium Print services, which are also targeted as the POSIX draft standard (1003.7.1), have been identified as the OSF DME base print services. Although Print Services was an original targeted DME Service, the availability of the Palladium Print Services directly from MIT and Palladium-based applications from third party vendors has led the OSF to refocus resources to other areas that cannot be as well addressed by third party sources and vendors.

F.2 DME Network Management Option (NMO)

The OSF DME Network Management Option (NMO) will focus on the management of enterprise devices and events which include activities, actors, collections, control, discovery, maps, modeling, monitoring, observation, and relationships. This option will support the incorporation of legacy systems by supporting existing network management protocols SNMP and CMIP in distributed environments. Bull, HP and SNI have contributed to the DME NMO, including X/Open's XMP API which serves as a transparency layer to the DME Instrument Request Broker (IRB). Figure F-2 details the NMO model within the DME architecture. It should be noted that only the center box of this figure is provided by the DME architecture. The NMO provides support for agents and third-party management applications through the SMP API and libraries. The NMO is integrated with DCE naming services and DME Event Services. Release of the OSF DME Network Management Option is expected mid-1994.

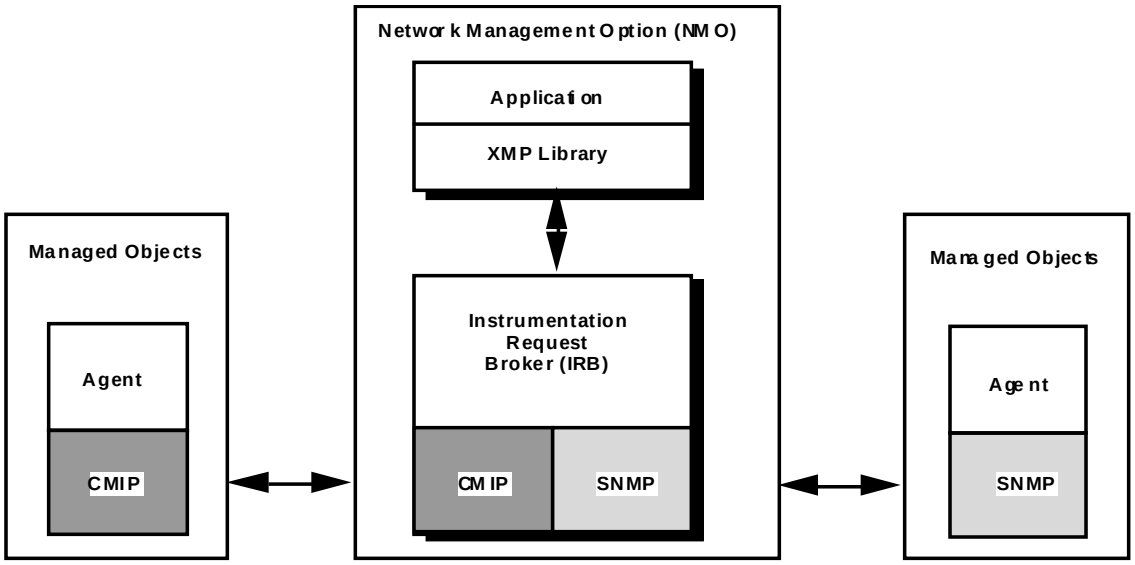


Figure F-2. DME Network Management Option (NMO)

F.3 DME Management Framework (MF) Object Services

The DME Object Services are provided by the DME Management Framework (MF). The Management Framework will provide support for object-oriented computing paradigms and supplies a consistent application programming interface (API) for interoperable distributed management applications.

Object services will support management of Object-related characteristics and entities: Object Life Cycle, Object Persistence, Object Notification, Object Naming, Object Security, as well as Time, Concurrency Control and Object Message Catalogs.

DME Object Services supports two Management Request Brokers (MRBs) which are central pieces in the Management Framework. The MRBs accept requests on objects, locate objects in the network and forwards requests. An asymmetric or traditional MRB is supported in addition to a symmetric or object-oriented MRB. The asymmetric MRB supports SNMP and CMIP management protocols and collects management information from devices in the network. The symmetric request broker uses DCE RPC mechanism, including the DCE Security Services.

Two object servers are provided. A short duration tasks object server is provided for such tasks such as password or attribute changes. A long duration task object server is provided for tasks such as network monitoring and configuration management.

The OSF DME will not only provide interoperability with devices/objects within DME but also with non-DME systems at several levels. The OSF DME Object Adapter will provide support for SNMP and CMIP, the OSI Structure of Management Information (SMI) and a common understanding of object definitions (OMG CORBA). Proprietary management systems may be

supported through the use of gateways. The OSF DME has specified an encapsulation technique to support this type of interoperability.

A major objective of the Object Services is CORBA compliance. Release of the OSF DME Object Services is expected late 1994 or early 1995. DME will not provide the CORBA ORB, but will interoperate with CORBA-compliant ORBs provided by third-party vendors, usually the operating system vendor. As indicated in the DCE Migration and Prototype Study, Object-Orientation and CORBA compliance are the major strategic objectives of most industry vendors, particularly in the POSIX workstation market.

F.4 DME Management User Interface

The OSF DME is based on OSF/Motif GUI. The following additional user interface components will be integrated within DME:

- HP OpenView Windows graphical mapping features
- High level Tivoli dialog language for object method user interaction specification
- Dialogue language to define screen layouts and actions on objects
- User interface definitions to objects stored in the managed object:

These objects will interpreted by DME Display Manager on request. The managed object includes attributes, operations, and associated user interface in a self-contained software module

- Several views of management information available: iconic, topological and management dialogs (command lines and dialog boxes).

F.5 DME Development Toolkit

DME framework services provide three APIs to management request brokers:

- Traditional MRB protocol stack API to SNMP and CMIP (X/Open XMP)
- ANSI-C based object-oriented API for management applications development
- C++ based object-oriented API for management applications development

These APIs will insulate the developer from the details of the management protocols and provide evolvability with minimal recoding. The DME Development Toolkit will provide high-level tools, including user interface development, event management support and object implementation.

Appendix G. Major Database Query Efforts Applicable to ECS

Numerous efforts are ongoing in parallel at defining advanced information system databases and database access. The query language is of interest to the communications sub-architecture in the area of how the actual query is mapped to the invoking communications service. Furthermore, the need to perform effective trend analysis with systems management data requires the use of object or multi-relational database systems. With the development of CORBA technology, a mapping of SQL to the CORBA IDL would be desirable. The following is an introduction to the current advanced query languages that have potential applicability to ECS - sections G3 through G6 are from the ECS Reference Model paper by Steve Carson.

G.1 ODMG

The Object Database Management Group (ODMG) has defined a standard method for accessing multivendor object databases. The specification allows object database vendors to develop independent implementations of object database technology, and provide the interchange of object-oriented database technology without affecting applications that use the OO database. The specification defines an object model, an object query language, object bindings or interfaces for C++ and Smalltalk OO programming languages, and an object definition language (ODL) based on the OMG IDL. ODMG has based most of its work on standards defined by the OMG and will continue to modify the ODMG specification to conform with evolving OMG standards. Eventually, ODMG will submit its specification to OMG for adoption as OMG's object querying, persistence, and replication services. ODMG vendors have declared intent to support the ODMG-93 specification by the end of 1994.

G.2 OGIS

The Open Geodata Interoperability Specification (OGIS) is an effort to develop geodata interoperability strategy and specification, organize a concensus process to guide development of the OGIS, and develop resources for OGIS implementation. The version 1 Draft 1 of the OGIS is scheduled for release on March 18, 1994, with a final draft 4 release of the OGIS scheduled for 9/15/94. The OGIS architecture relies on distributed object technology, providing interoperability to geoprocessing services, transfer format access managers, conventional applications, GIS DB Access managers, transformation services, distribution format access managers, legacy applications and Open GIS applications. A virtual geodata model (VGM) working group is in the process of developing an object-oriented model of geographic information, providing a link between private data stores and client applications of spatial (geometric), temporal, thematic, associative, and metadata attributes. A VGM prototype is in work by OGIS to describe geodata object's attributes using SDTS and SAIF, defining behaviors using OMG IDL and SGL3. The test system is the Postgres database management system. The OGIS will include the specification of a framework. The objectives of the framework are to support tool communications, support data

exchange, support interoperable tools, support multiple layers of access, leverage existing tools, support other external (COTS/GOTS) tools, and support distributed object environments. OGIS is strongly considering adapting the OGIS within the common object integration framework of OMG CORBA. Current proposed framework services of the OGIS have a strong parallel to the major services of ECS SDPS and include the following:

- User Interface Services:

  provides coherent and consistent UI for all tools, separates presentation of functionality from its provision, involves all aspects of the framework

- Data Management Services:

  supports persistent object storage and access, object links and relationships, object naming, composite objects, transaction control, and meta-data

- Data Integration Services:

  query services, meta-data services, data-driven triggers, views, work spaces and object distribution

- Process/Task Management Services:

  tool configuration, tool synchronization, task descriptions, process management, event-driven triggers, transaction control

- Message Services:

  tool registration, message broadcast and delivery, object distribution, and object synchronization

Interface mechanisms to OGIS include the use of IDLs, APIs and the CORBA proposed Dynamic Invocation Interface (DII). The basic services of the OGIS framework may use MITRE's DISCUS project as a starting point. These services are being considered by MITRE for submission to the forthcoming OMG data interchange and query object services request for specifications. These services are detailed below:

- Exchange:

  data objects wrapped in a generic OGIS package which are exchanged and unwrapped by receiving application

- Convert:

  generic data format conversion/translation service from/to OGIS VGM

- Query:

  supports dynamic queries and metadata browsing

- Execute:

  encapsulation services for leveraging COTS/GOTS applications

G.3 SQL3

The ANSI X3H2 Technical Committee on Database Language SQL is in the process of defining and SQL3 object model with a target 1995 or 1996 availability date.

National and international SQL standardization committees are now focusing on development of future extensions for meeting the stated requirements of managing complex objects in engineering and multimedia environments. These extensions include object identifiers, abstract data types, inheritance hierarchies, and all of the other features normally associated with object data management.

This second substantial enhancement (ISO/IEC Project 1.21.3.4). often called SQL3, is currently at the working draft stage with standardization expected in the 1996 time frame. The SQL3 specification is a forward-looking SQL enhancement that intends to provide a computationally complete language for defining and managing persistent objects. SQL3 also contains Triggers and Assertion that can form the basis of "intelligent" database management systems.

An important feature in the SQL3 specifications is known as object oriented extensions of the SQL language including user-defined, abstract data types (ADT), including methods, object identifiers, subtypes and inheritance, polymorphism, type templates, and integration with existing facilities. The base data types in SQL3 include fixed-length and variable-length character strings, fixed-length and variable-length bit strings, fixed and floating point numerics, dates, times, time stamps, intervals, Booleans, and enumerations.

At the Application Integration Architectures Workshop (NIST, December 1993) Object Model Soup Birds-of-a-Feather Session, it was recognized that providing matrixes of object model features could lead to better and perhaps standard definitions of object model features. This approach would enable a way to compose object models from OMG Component parts, establishing Profiles for SQL3.

G.4 SQL/CLI

An emerging standard for an SQL call level interface (SQL/CLI) is under development in ISO with Draft International Standard (DIS) status expected sometime during calendar year 1994. The call level interface is a requirement for third-party software developers who produce "shrink-wrapped" software for use on personal computers and workstations. They do not wish to use a Module processor or an Embedded SQL preprocessor binding style because they do not wish to distribute any source code with the products they sell to individual users. Instead they desire a services call interface to SQL data repositories that can be invoked from the calling environment provided by the host operating system. The calls to the SQL data repository can then be embedded in the object code just like calls to any other system service.

The Call Level Interface is an alternative mechanism for executing SQL statements. SQL/CLI consists of a number of functions that:

1) allocate and deallocate resources;
2) control connections to SQL-servers;

3) execute SQL statements using mechanisms similar to Dynamic SQL;

4) obtain diagnostic information; and

5) control transactions.

G.5 SQL/ERI

The emerging SQL External Repository Interface (SQL/ERI) standard supports integration of heterogeneous data repositories to provide user access to all forms of data while retaining full use of the SQL language.

An SQL/ERI Server may provide an SQL Call Level Interface binding style according to the requirements of the emerging standard for SQL/CLI. The SQL/CLI specification should reach a stable state in the ISO/IEC standardization process during calendar year 1994.

G.6 SQL/MM

This material is based on the ISO Working Draft text of the SQL Multimedia and Application Packages (SQL/MM) text and on the work plan for that work (ISO/IEC JTC1/SC21 N ????.) [Note: This material is still informal at this point since the SQL/MM work as yet has no formal definitions and there is not sufficient time to develop one for this draft of this document.]

The SQL Multimedia and Application Packages (SQL/MM) project will specify packages of SQL abstract data types for use in various areas. The initial work focuses on three areas where support is urgently needed:

1. Foundation (Part 1),
2. Full Text (Part 2), and
3. Spatial. (Part 3).

Future packages (i.e. additional parts of the standard) are expected to include:

4. Still Graphics,
5. Still Images,
6. Animation,
7. Full Motion Video,
8. Audio,
9. Euclidean Geometry,
10. Seismic,
11. Geography,
12. Music, and
13. Mathematical Structures.

Each part will specify a package of abstract data type (ADT) definitions using facilities for ADT definition provided in the Database Language SQL enhancement work, SQL3 (ISO/IEC Project 1.21.3.4). Other ISO/IEC and CCITT standards will be used as the basis of definition for SQL ADTs and each part will provide transformations to and from standard external representations whenever appropriate.

The main intent of this work is to allow SQL applications to use the same ADTs across different applications areas, thereby promoting interoperability and sharing of data, and to encourage performance optimization over a manageable collection of types. It is best if many of the needed packages are developed under a single coordinated effort to avoid duplication or incompatible specification of the more elementary ADTs that get used in a number of different application areas.

[Note: Little formal work has been done to insure that the ADTs defined here will interwork with active objects defined according to OMG technology and with the Persistent Storage Manager being defined within OMG as part of the JOSS specification.]

Appendix H. Major User and Vendor Driven Consortia Applicable to ECS

The Report on the Application Integration Architectures Workshop (NIST, December 1993) provides detailed information of high value in understanding the roles and charters of many standards and industrial consortia activities in the software architectural development. A common framework for understanding the major activities of the standards development organizations (SDOs) and the consortia is shown below.

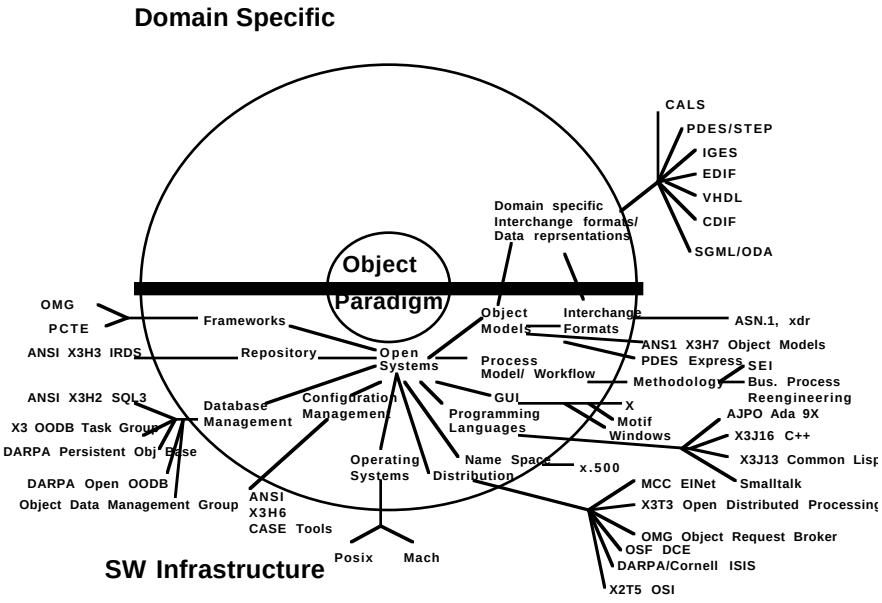


Figure H-1. SDO and Consortia Software Activities

Understanding major consortia, their roles and inter-relationships is important to effect an up-to-date knowledge of activities to be tracking, collaborating, and/or contributing.

H.1 X/Open

The mission of X/Open is to bring users greater value from computing through the practical implementation of open systems. X/Open uses the OMG object model, and essentially 'brands' technology solutions for open systems users. Key technologies 'branded' by X/Open generally include consortia approved technologies that have undergone a series of technical solicitation and evaluations. Recent X/Open certifications include the Open Software Foundations DCE, the OMG CORBA, and most recently, the Tivoli Management Environment (TME) for distributed systems management.

H.2 OMG

The mission of the Object Management Group (OMG) is to promote cross-platform interoperability using object technology. Key functions of the OMG are to identify problems associated with the fulfillment of their mission, place requests for specifications, evaluate the submissions, select an appropriate solution, and publish the specification for independent vendor implementation.

H.3 COSE

The Common Open System Environment is a consortium that works to identify technology problems, evaluate current technology solutions, and make recommendations/selections of technologies to resolve technology problems.

H.4 OSF

The mission of the Open Software Foundation is to provide open systems solutions for the advancement of distributed processing. Key functions of the OSF are to identify problems associated with the fulfillment of their mission, place requests for technology, evaluate the submissions, select an appropriate solution, integrate the solution, QA the solution, then make the solution generally available for vendor implementation. The specific functions of the OSF may change in light of ongoing negotiations between COSE and OSF. An announcement will be made at UniForum; current speculation is that OSF will become a technology integrator, leaving the technology analysis and selection process to COSE.

H.5 NMF

The mission of the Network Management Forum (NMF) is to accelerate the availability of management solutions for networked information systems. NMF is primarily based on the OSI Management Information Model (ISO/IEC 10165-1), although object model reconciliation with the OMG object model has been investigated. NMF is preparing for release in 4Q94 a set of implementation agreements and specifications for the management of networked information systems entitled "OMNIPoint 2'. NMF has partner relationships with X/Open, OSF, OMG, UI, NIST, and CCTA.

H.6 ATM Forum

TBS

H.7 CIL

TBS

Abbreviations and Acronyms

ADC	Affiliated Data Center
ADGE	Air Defense Ground Environment
AI	artificial intelligence
ANSI	American National Standards Institute
API	application programmer's interface
ASCII	American Standard Code for Information Interchange
ASTER	Advanced Space borne Thermal Emission and Reflection Radiometer
AVHRR	Advanced Very High Resolution Radiometer
CEOS	Committee on Earth Observation Satellites
CINTEX	CEOS Inventory Interoperability Experiment
CORBA	Common Object Request Brokering Architecture
COTS	commercial off-the-shelf (hardware or software)
CSMS	Communications and System Management Segment
DAAC	Distributed Active Archive Center
DBMS	data base management system
DCE	Distributed Communications Environment of OSF
E-mail	electronic mail
ECS	EOSDIS core system
EOS	Earth Observing System
EOSDIS	EOS Data and Information System
ESA	European Space Agency
ESDIS	Earth Science Data and Information System
ESIS	European Space Information System
FIFE	First ISCLSCP Field Experiment
FITS	Flexible Image Transfer System
FOS	Flight Operations Segment
GC	Global Change
GCDIS	Global Change Data Information System

GCRP	Global Change Research Program
GIF	Graphical Interchange Format
GSFC	Goddard Space Flight Center
GUI	Graphic User Interface
H/W	hardware
HITC	Hughes Information Technology Corporation
HDF	hierarchical data format
HMI	human machine interface
IDL	Interface Definition Language
IEEE	Institute of Electrical and Electronic Engineers
IMS	Information Management System (ECS)
IP	International Participant/Partner
ISO	International Standards Organization
JGOFS	Joint Global Ocean Forecasting System
L0-L4	Level 0 through level 4 (processing)
LaRC	Langley Research Center
kbytes	Kilo-bytes (10^3)
Mbytes	Mega-bytes (10^6)
MoU	memorandum of understanding
MTPE	Mission to Planet Earth
NASA	National Aeronautics and Space Administration
NCSA	National Center for Supercomputer Applications
netCDF	network version of the Common Data Format
NIIT	National Information Infrastructure (NII) Testbed
NRC	National Research Council
NRT	near real time (data)
OODBMS	object oriented database management system
ODC	other data center
ODL	Object Description Language
OS	operating system
OSF	Open Systems Foundation

OSI	Open System Interconnect
PI	principal investigator
POSIX	Portable Operating System Interface for Computer Environments
QA	quality assurance or quality assessment
RDBMS	relational database management system
RFQ	request for quote
RPC	remote procedure call
S2000	Sequoia 2000 project
S/W	software
SCF	Science Computing Facility
SDPS	Science Data Processing Segment
SFDU	Standard Formatted Data Unit
SMC	System Management Center (ECS)
SQL	Structured Query Language
UIT	User Interface Terminal (ESA)
UNIDATA	University Data System (NSF - Atmospheric Sciences Division)
UserDIS	Data Information System based on providers in the earth science user community
V0	Version Zero (EOSDIS)
WAIS	Wide Area Information System
WOCE	World Ocean Circulation Experiment
WWW	World Wide Web
X.500	OSI standard for directory services
XBT	Instrument for measuring temperature and salinity profiles with depth in the ocean
X^nDIS	an architectural concept to satisfy EOSDIS, GCDIS and UserDIS needs

Bibliography

Reference models

1. Final text to ISO 10032 - Reference Model for Data Management
2. ISO/JTC1/SC21 N 8228 - Revised test of DIS 7498-1, Information Technology - Open Systems Interconnection - Basic Reference Model, Nov. 1993, Second Edition

Open Distributed Processing

1. ISO/JTC1/SC21 N 8218 - Working Draft for Information Technology - Open Distributed Processing - Basic Reference Model of Open Distributed Processing, Part 1: Overview and Guide to Use, September 1993.
2. Committee Draft ISO/IEC CD 10746-3.2; ISO/JTC1/SC21 N 8125 - Information Technology - Open Distributed Processing - Part 3: Systems Interconnection - Prescriptive Model, Dec. 1993.
3. ISO/JTC1/SC21 N 8034 - Liason Statement to the Object Management Group, Aug. 1993.
4. ISO/IEC JTC1/SC21 N - Project JTC1.21.59. Rec. X.xxx/Draft ODP Trading Function ISO/IEC xxx: 199x/ Draft ODP Trading Function, 12 November, 1993.

Object Models

"Comparision of the OMG and ISO/CCITT Object Models", Report of the Joint NM Forum/OMG Taskforce on Object Modeling, February 1993 .

OMG CORBA

1. OMG TC Document 93.7.2 - Object Request Broker Architecture; Version 0.0; July 18, 1993

OMG Object Services

1. OMG Document 92.8.1 - Object Services Architecture, Revised August 28, 1992.
2. OMG Document 92.8.5 - Object Service Roadmap, Revision 1.0, 28 August 1992.
2. OMG TC Document 93.7.1 - Joint Object Services Submission: Submission Overview, Revised July 3, 1993.
3. OMG TC Document 93.3.2 - Joint Object Services Submission: Naming Service Specification, Revised May 14, 1993.
4. OMG TC Document 93.7.3 - Joint Object Services Submission: Event Service Specification, Revised July 2, 1993.
5. OMG TC Document 93.7.4 - Joint Object Services Submission: Life Cycle Services Specification, Revised July 2, 1993.

6. OMG TC Document 93.2.4 - Joint Object Services Submission: Life Cycle and Association Services Specification, Feb. 19, 1993.

7. OMG TC Document 93.5.3 - Joint Object Services Submission: PSM Specification, Revised May 14, 1993

8. Common Object Services Specification, Volume I, Revision 1.0, March 1, 1994.

Database

1. ISO/JTC1/SC21/WG3 N 1614 - ISO Working Draft SQL Multimedia and Application Packages (SQL/MM) Part 3: Spatial, Sept. 1993

2. ISO/JTC1/SC21 N 7689 - Revised Text of DIS 9579-1, Information Technology - Remote Database Access - Part 1: Generic Model, Service and Protocol

3. ISO/JTC1/SC21 N 7596 - Working Draft SQL Call Level Interface (SQL/CLI, Jan. 1993)

4. ISO/JTC1/SC21 N 7597 - Working Draft - Persistent SQL Modules, Jan. 1993.

5. ISO/JTC1/SC21 N 7703 - Revised Text of DIS 9579-2, Information Technology - Remote Database Access - Part 2: SQL Specification, Mar. 1993.

General OSI

1. ISO/IEC/JTC1/SC21 N 6341 - Revised Text of CD 10731, Information Technology - Open Systems Interconnection - Conventions for the Definition of OSI Services, Aug. 1993.

2. ISO/IEC/JTC1/SC21 N 7851 - Final Text of ISO 9545:1993; - Information Technology - Open Systems Interconnection - Application Layer Structure, May 1993.

3. ISO/IEC/JTC1/SC21 N 7669 - Text of ISO/IEC DIS 9072-1, Information Technology - Remote Operations - Part 1: Concepts, Model, and Notation, Mar. 1993.

RO and RPC

1. ISO/JTC1/SC21 N 8212 - Revised Text of CD 11578-1.2, Information Technology - Open Systems Interconnection - Remote Procedure Call - Part 1: Model, Sept. 1993.

2. ISO/JTC1/SC21 N 7670 - Text of ISO/IEC DIS 9072-2, Information Technology - Remote Operations - Part 2: OSI Realizations - Remote Operations Service Element (ROSE) Service Definition, Mar. 1993.

3. ISO/JTC1/SC21 N 7671 - Text of ISO/IEC DIS 9072-2, Information Technology - Remote Operations - Part 3: OSI Realizations - Remote Operations Service Element (ROSE) Protocol Specification, Mar. 1993.

Network Management Standards and Related Documentation

RFC 1157 - "A Simple Network Management Protocol" J.D. Case, May 1990

RFC 1155 - "Structure and Identification of Management Information for TCP/IP-based Internets", M. Rose and K. McCloughrie, May 1990

RFC 1189 - "Common Management Information Service and Protocol over TCP/IP (CMOT)", K. McCloughrie and M. Rose, March 1991

RFC 1213 - "Management Information Base for Network Management of TCP/IP-base Internets:

MIB-II", K. McCloughrie and M. Rose, March 1991

RFC XXX - "Introduction to the Simple Network Mangement Protocol Framework", J.D. Case

ISO/IEC IS 10733 Information Technology - Telecommunications and information exchange between systems - Elements of Management Information Related to OSI Network Layer Standards, approved July 1992

ISO/IEC IS 10737 Information Technology - Telecommunications and information exchange between systems - Elements of Management Information Related to OSI Transport Layer Standards, approved July 1992

M.T. Rose, The Simple Book - An Introduction to Management of TCP/IP-based Internets, Prentice Hall, 1990. (ISBN 0-13-812611-9)

M.T. Rose, The Open Book, A Practical Perspective on OSI, Prentice Hall, 1990.

Network Management Forum: Forum 006, "Forum Library - Volume 4: OMNI*Point* 1 Definitions", Issue 1.0, August 1992

P802.1B/D20 or later, Draft Standard 802.1B: LAN/MAN Management, Jan. 27, 1992

P802.3.K/D10 or later, Draft Supplement to ANSI/IEEE Std. 802.3 - 1992 Edition, Repeater Management, July 11, 1992

CCITT Draft Recommendation M.1300, Generic Network Information Model, November, 1991

OSI Network Management Standards (NIST-GNMP related, DME-OSI related

A. General

NIST Special Publications 500-202, "Stable Implementation Agreements for Open Systems Interconnection Protocols, Version 5 Edition 1"

NIST (GNMP Users Guide to version 1 GNMP)

Management Information Catalog, Issue 1.0, June 1992; jointly published by NIST, the OIW NMSIG, and NMF

A. Architecture and Organization of Management

ISO/IEC IS 7498-4 "Information Technology - Open Systems Interconnection - Basic Reference Model - Part 4: Management Framework", 1989

ISO/IEC IS 10040 (CCITT Rec. X.701) "Information Technology - Open Systems Interconnection - Systems Management Overview", 1991

B. Communication of Mangement Information

ISO/IEC IS 9595 (CCITT Rec. X. 710) "Information Technology - Open Systems Interconnection - Management Information Service Specification - Common Management Information Services Definition (CMIS), 1991

ISO/IEC IS 9596 (CCITT Rec. X. 711) "Information Technology - Open Systems Interconnection - Management Information Protocol Specification - Common Management Information Protocol Specification (CMIP), 1991

C. Structure of Management Information

ISO/IEC IS 10165-1 (CCITT Rec. X.730) "Information Technology - Open Systems Interconnection, Management Information Services - Structure of Management Information - Part 1: Management Information Model", 1991

ISO/IEC IS 10165-2 (CCITT Rec. X.721) "Information Technology - Open Systems Interconnection, Management Information Services - Structure of Management Information - Part 2: Definitions of Management Information", 1991

ISO/IEC IS 10165-4 (CCITT Rec. X.723) "Information Technology - Open Systems Interconnection, Management Information Services - Structure of Management Information - Part 4: Guidelines for the Definition of Managed Objects", 1991

ISO/IEC DIS 10165-5 "Information Technology - Open Systems Interconnection, Management Information Services - Structure of Management Information - Part 5: Generic Management Information", February 1992

ISO/IEC DIS 10165-6 "Information Technology - Open Systems Interconnection, Management Information Services - Structure of Management Information - Part 6: Requirements and Guidelines for Implementation Conformance Statement Proformas Associated with Management Information", 21 February 1992

D. System Management Functions

ISO/IEC IS 10164-1 (CCITT Rec. X730) "Information Technology - Open Systems Interconnection - Systems Management - Part 1: Object Management Function (OMF)", 1991

ISO/IEC IS 10164-2 (CCITT Rec. X731) "Information Technology - Open Systems Interconnection - Systems Management - Part 2: State Management Function (SMF)", 1991

ISO/IEC IS 10164-3 (CCITT Rec. X732) "Information Technology - Open Systems Interconnection - Systems Management - Part 3: Attributes for Representing Relationships (ARR)", 1991

ISO/IEC IS 10164-4 (CCITT Rec. X733) "Information Technology - Open Systems Interconnection - Systems Management - Part 4: Alarm Reporting Function (ARF)", 1991

ISO/IEC IS 10164-5 (CCITT Rec. X734) "Information Technology - Open Systems Interconnection - Systems Management - Part 5: Event Report Management Function (ERMF)", 1991

ISO/IEC IS 10164-6 (CCITT Rec. X735) "Information Technology - Open Systems Interconnection - Systems Management - Part 6: Log Control Function (LCF)", 1991

ISO/IEC IS 10164-7 (CCITT Rec. X736) "Information Technology - Open Systems Interconnection - Systems Management - Part 7: Security Alarm Reporting Function (SARF)", 1991

FUTURE - Workload Monitoring Function

FUTURE - Summarization Function

FUTURE - Objects and Attributes for Access Control

FUTURE - Security Audit Trail Function

FUTURE - Accounting Metering Function

FUTURE - Test Management Function

FUTURE - Changeover Function

FUTURE - General Relationship Model Function

FUTURE - Management Domain Function

FUTURE - Management Knowledge Management Function

FUTURE - Response Time Monitoring Function

FUTURE - Scheduling Function

FUTURE - Time Management Function

E. Security Functions

NIST FIPS Pub. 112: "Password Usage", May 1985

NIST proposed FIPS: "Specifications for a Secure Has Standard", January 22, 1992

ISO/IEC IS 9594-3 "Information Technology - Open Systems Interconnection - The Directory - Part 3: Abstract Service Definition", 1988

ISO/IEC IS 9594-8 "Information Technology - Open Systems Interconnection - The Directory - Part 8: Authentication Framework", 1988

ISO/IEC IS 8649 Amdl 1: "Information Technology - Open SystemsInterconnection - Service Definition for the Association Control Service Element, Amendment 1: Peer-entity Authentication during Association Establishment", 1990

ISO/IEC IS 8650 Amdl 1: "Information Technology - Open Systems Interconnection - Service Definition for the Association Control Service Element, Amendment 1: Peer-entity Authentication during Association Establishment", 1990

OMNIPoint Point Paper - Keith Willetts, NMF